Design and Implementation of a Natural User Interface to support the annotation process for multimedia learning

Relatore: Prof. Licia Sbattella
Correlatore: Ing. Roberto Tedesco

Thesis by:
Diogo Borges Krobath, 764424

Academic Year 2012/2013
Abstract

Lectures based on Power Point slides are widely diffuse nowadays in universities. The use of such method facilitates displaying diverse formats of multimedia content such as images, audio and video. However, in the other hand, it brings challenges to professors to engage student’s participation mainly due the passive situation in which students are in this approach and often lack of interactivity student-professor.

This thesis work explores the cognitive aspects for multimedia learning and the nature of the notes taken by students in order to understand the main issues dealt by them and consequently collect requirements for a software solution to support the annotation process for multimedia learning in a classroom.

This work continues by introducing the paradigm of Natural User Interfaces and suggests properties of this paradigm, such as immersiveness to the content and increase in interactivity, as solutions to engage actively students with the lecture content and consequently contribute for enhancing the multimedia learning.

Furthermore recent technologic outcomes are introduced to support the design and development on Natural User Interfaces paradigm such as Windows 8 and the applicatives on Windows Store running on this platform. The process of design uses all the requirements identified in previous analysis and adequate them to the standard navigation, commanding and gestures patterns defined for Windows Store App.

Polinotes Student App is the result of the software implementation. It is compatible with Windows App Store and acts as a client able to receive multimedia content from Power Point slides (through the connectivity with Polinotes package) and at the same time contains the basic resources for supporting the annotation process of a student together with the multimedia content just brought.
Sommario

Lezioni basate su diapositive Power Point sono ampiamente diffuse al giorno d'oggi nelle università. L'utilizzo di questo metodo facilita la visualizzazione di diversi formati di contenuti multimediali come immagini, audio e video. Comunque, questo porta sfide per docenti di coinvolgere la partecipazione dello studente principalmente a causa della situazione passiva in cui gli studenti sono in questo approccio e spesso mancano di interattività studente-professore.

Questo lavoro di tesi esplora gli aspetti cognitivi di apprendimento multimediali e la natura degli appunti presi dagli studenti, al fine di comprendere le principali questioni trattate da loro e di conseguenza raccogliere i requisiti di una soluzione software per supportare il processo di annotazione per l'apprendimento multimediale in una classe.

Questo lavoro continua con l'introduzione del paradigma delle Natural User Interfaces e propone proprietà di questo paradigma, come immersività al contenuto e aumento di interattività, come soluzioni per coinvolgere attivamente gli studenti con il contenuto di lezione e di conseguenza contribuire per migliorare l'apprendimento multimediale.

Inoltre, recenti risultati tecnologici vengono introdotti a supporto della progettazione e dello sviluppo sul paradigma Natural User Interfaces, come Windows 8 e gli applicativi su Windows Store di questa piattaforma. Il processo di progettazione si avvale di tutti i requisiti individuati nella precedente analisi e adeguata a standard navigazione, comando e modelli di gesti definiti per Windows App Store.

Polinotes Student App è il software risultato dell'implementazione. È compatibile con Windows Store App e agisce come un client in grado di ricevere contenuti multimediali da diapositive Power Point (tramite la connettività con il pacchetto Polinotes) e al tempo stesso contiene le risorse di base per sostenere il processo di annotazione di uno studente con la multimedialità contenuto appena portato.
Acknowledges

I would like to thank my project supervisor, Prof. Licia Sbattella for the opportunity to work on this very interesting thesis project as well as the resources provided to support my work such as a Tablet-PC and books. Also a many thanks for my project advisor, Ing. Roberto Tedesco, for his continuous guidance, support and great ideas throughout the project.

I also have to thank particularly my parents during this period of studies in Italy, their motivation and support were fundamental for the completion of this thesis.

Besides that, thanks also for my manager at Novartis, Andrew Garrood for letting me dedicate a few days from my working activities for developing this thesis.
## Contents

1. Introduction ........................................................................................................................................... 1  
   1.1. The Project CATS ......................................................................................................................... 1  
   1.2. Motivations ..................................................................................................................................... 2  
   1.3. Objectives ....................................................................................................................................... 3  
   1.4. Thesis organization ......................................................................................................................... 3  
2. Multimedia learning in classroom ............................................................................................................. 5  
   2.1. Multimedia learning ....................................................................................................................... 5  
   2.2. Learning in classroom environment ............................................................................................... 6  
   2.3. Methods for presenting content in classroom ................................................................................. 6  
   2.4. The multimedia content .................................................................................................................. 8  
   2.5. Cognitive processes for multimedia learning .................................................................................. 9  
      2.5.1. External cognition .................................................................................................................... 9  
      2.5.2. Cognitive Theory of Multimedia Learning ............................................................................. 10  
   2.6. Learning principles ......................................................................................................................... 13  
   2.7. Conclusions for a software solution to support multimedia learning ........................................... 14  
3. Process of annotation .............................................................................................................................. 16  
   3.1. The process of annotation ............................................................................................................. 16  
   3.2. Annotation for multimedia learning ............................................................................................... 17  
   3.3. Properties of class notes ............................................................................................................... 18  
   3.4. Methods for annotation in classes .................................................................................................. 19  
      3.4.1. Pen and paper ........................................................................................................................... 22  
      3.4.2. Computer based annotation .................................................................................................... 22  
   3.5. The tasks of annotation process in classes ..................................................................................... 29  
      3.5.1. Pen and paper ........................................................................................................................... 30
3.5.2. Computer based annotations .............................................................................. 32
3.6. Conclusions for improving process of annotation .................................................. 34
4. Natural user interfaces .............................................................................................. 37
  4.1. The evolution of interfaces ..................................................................................... 37
  4.2. Natural User Interfaces (NUI) ............................................................................... 39
  4.3. NUI for learning .................................................................................................... 40
  4.4. Windows 8 as a NUI ............................................................................................. 43
  4.5. Conclusions ........................................................................................................... 48
5. Designing for the applicative ..................................................................................... 51
  5.1. Design for Windows Store Apps ........................................................................... 51
         5.1.1. Navigation Patterns .................................................................................... 51
         5.1.2. Commanding patterns ............................................................................... 54
         5.1.3. Touch interaction patterns .......................................................................... 57
         5.1.4. Touch posture .............................................................................................. 60
         5.1.5. Touch targets ............................................................................................... 63
  5.2. Basic actions for the design of annotation process ............................................... 65
         5.2.1. Handwriting ................................................................................................ 65
         5.2.2. Text ............................................................................................................. 66
         5.2.3. Illustration .................................................................................................. 67
         5.2.4. Page ............................................................................................................ 67
  5.3. Polinotes Student App design ............................................................................... 68
         5.3.1. Gestures design .......................................................................................... 68
         5.3.2. Interface modes .......................................................................................... 74
         5.3.3. Commanding design .................................................................................. 75
         5.3.4. Navigation design ........................................................................................ 76
6. Polinotes Student App implementation ...................................................................... 77
  6.1. Polinotes ............................................................................................................... 77
         6.1.1. The choice for Polinotes .............................................................................. 77
         6.1.2. Components of Polinotes Package ............................................................... 79
         6.1.3. Connectivity pattern of Polinotes ................................................................ 79
6.1.4. Complete usage scenario ................................................................. 81
6.1.5. Technology for connectivity .............................................................. 85
  6.1.5.1. Windows Communication Foundation ........................................... 85
  6.1.5.2. WCF implementation on Polinotes Teacher .................................. 86
  6.1.5.3. WCF implementation on Polinotes Student .................................. 88
  6.1.5.4. WCF implementation on Polinotes Discovery .............................. 90
6.2. Development of Polinotes Student App .................................................. 91
  6.2.1. Objectives ......................................................................................... 91
  6.2.2. Technologies for development .......................................................... 92
  6.2.3. Software architecture ........................................................................ 94
  6.2.4. Implementation for Windows Store environment ............................ 98
  6.2.5. Implementation of elements for the annotation process .................. 100
  6.2.6. Implementation of gestures ............................................................ 105
    6.2.6.1. Manipulation events ................................................................. 106
    6.2.6.2. Pointer events ........................................................................... 108
    6.2.6.3. Static gesture event ................................................................. 111
6.2.6. Implementation of gestures ............................................................ 105
7. Conclusions and future work .................................................................... 113
# List of Figures

Figure 1 - Components of learning.......................................................................................... 6  
Figure 2 - Cognitive Theory of Multimedia Learning.............................................................. 11  
Figure 3 - Methodologies for taking notes............................................................................... 20  
Figure 4 - Advantages of paper and electronic notes .............................................................. 21  
Figure 5 - Microsoft One Note............................................................................................... 24  
Figure 6 - LiveNotes............................................................................................................. 25  
Figure 7 - NotePals interface................................................................................................ 26  
Figure 8 - StuPad interface.................................................................................................... 27  
Figure 9 - Windows Reader interface ................................................................................... 28  
Figure 10 - One Note App Interface ..................................................................................... 29  
Figure 11 - Evolution of the interfaces paradigm ................................................................. 39  
Figure 12 - Windows 8 Start screen....................................................................................... 44  
Figure 13 - Using App contracts on Windows 8 .................................................................... 44  
Figure 14 - App with tiles features ...................................................................................... 45  
Figure 15 - Semantic Zoom.................................................................................................. 46  
Figure 16 - Defining App setting over Charms ..................................................................... 47  
Figure 17 - Hierarchical navigation pattern ........................................................................ 52  
Figure 18 - Hub, Section and Details pages ......................................................................... 53  
Figure 19 - Flat navigation pattern ...................................................................................... 53  
Figure 20 - Navigation using the flat pattern ...................................................................... 54  
Figure 21 - Canvas area ........................................................................................................ 55  
Figure 22 – Enabling navigation and command app bars ..................................................... 55  
Figure 23 - Command to enable the Charms ..................................................................... 56  
Figure 24 - Enabling the App bar ........................................................................................... 57  
Figure 25 - Activating content menu .................................................................................... 57  
Figure 26 - Press and hold to learn ...................................................................................... 58  
Figure 27 - Tap for primary action ...................................................................................... 58  
Figure 28 - Slide to pan ......................................................................................................... 58  
Figure 29 - Swipe to select, command and move ................................................................... 59  
Figure 30 - Pinch to stretch and zoom ................................................................................. 59  
Figure 31 - Turn to rotate ...................................................................................................... 59  
Figure 32 - Interaction areas ................................................................................................. 60  
Figure 33 - Reading areas .................................................................................................... 61
Figure 34 - One hand holding, one hand interacting with light to medium interaction .......... 61
Figure 35 - Two hands holding, thumbs interacting with light to medium interaction .......... 62
Figure 36 - Device rests on table or legs, two hands interacting with light to heavy interaction. 62
Figure 37 - Device rests on table or stand, with or without interaction .......................... 62
Figure 38 - Right hand interaction .................................................................................. 63
Figure 39 - Minimum size recommended for touch targets ............................................. 63
Figure 40 - Touch size for critical operations ................................................................. 64
Figure 41 - Minimum touch target .................................................................................... 64
Figure 42 - Relationship between target size and missed taps ........................................... 64
Figure 43 - Stages of a gesture ......................................................................................... 69
Figure 44 - Polinotes concept ........................................................................................... 78
Figure 45 - Publisher and subscriber pattern ..................................................................... 80
Figure 46 - Implementation of publisher-subscriber pattern on Polinotes ......................... 81
Figure 47 - Polinotes usage steps ...................................................................................... 82
Figure 48 - Process of connection among components ..................................................... 83
Figure 49 - Polinotes sending multimedia content to students ......................................... 84
Figure 50 – Software architecture of Polinotes Teacher .................................................... 87
Figure 51 - Software architecture of Polinotes Student ..................................................... 89
Figure 52 - Software architecture of Polinotes Discovery .................................................. 90
Figure 53 - Windows Store App API ................................................................................ 92
Figure 54 - Microsoft Visual Studio 2012 ........................................................................ 94
Figure 55 - Software architecture of the app ................................................................... 97
Figure 56 - Canvas implementation ................................................................................ 98
Figure 57 - Navigation bar implementation .................................................................... 99
Figure 58 - Command bar implementation ..................................................................... 99
Figure 59 - Images placed on Canvas ............................................................................. 101
Figure 60 - Two groups of handwriting texts defined (red and black) by timer trick........ 102
Figure 61 - Bounding box trick with tittle mark .............................................................. 103
Figure 62 - Manipulating object text ............................................................................... 104
Figure 63 - Inside rectangle gesture ............................................................................... 110
Figure 64 - Gesture create space .................................................................................... 111
List of tables

Table 1 - Comparison between annotation over empty pages vs. pages with printed content... 31
Table 2 - Comparison between annotation with keyboard and mouse vs. multi-touch and pen interactions........................................................................................................... 34
Table 3- Actions and elements of annotation ........................................................................ 68
Table 4- Rotate gesture ........................................................................................................... 69
Table 5 - Resize gesture ........................................................................................................... 70
Table 6 - Move gesture ........................................................................................................... 70
Table 7 - Change content gesture ................................................................................................ 71
Table 8 - Select part of content ................................................................................................ 71
Table 9 - Next/previous gestures ............................................................................................ 71
Table 10 - Selection inside rectangle gesture ........................................................................... 72
Table 11 - Tap to select gesture .............................................................................................. 72
Table 12 - Change distance among objects ............................................................................. 73
Table 13 - Create/ Reduce space gesture .................................................................................. 74
Table 14 - Interface modes ...................................................................................................... 75
Table 15 - WinRT API events .................................................................................................. 106
Table 16 - Static gestures implemented ................................................................................... 112
1. Introduction

1.1. The Project CATS

The project developed in this thesis work is part of the Campus Tools for Students (CATS) project, which is an initiative composed by three Italian universities: Politecnico di Milano, the responsible for creating the initiative, Università degli Studi di Milano Bicocca and Università degli Studi di Modena e Reggio Emilia.

The primary objective of the CATS project is to promote the inclusion of students with some hearing impairment or dyslexia problems in the university classroom, therefore minimizing the abandonment of the courses and improving their performance. Students with such impairments are commonly reported for having significant difficulties to follow lessons, study and retrieval of course materials; in particular, students face considerable difficulties along lectures, especially on those based on multimedia content from slides, in which they present inabilities to follow the pace of a lesson and take notes at the same time.

Therefore, by taking advantage of the new technologies and the current wireless coverage of the universities, the CATS project aims to:

- create and test innovative software tools to support the use of in educational activities and study;
- make accessible, storable, exchangeable and able to be processed immediately, multimodal, integrated and editable the multimedia content presented in classroom by professors;
- create services for software virtualization aimed at disability, to mainly allow their use without installment requirements on the machine of the student;
- develop software for the analysis of teaching materials and their transformation into "materials with improved accessibility."
The CATS project represents an important contribution to the Italian University with respect to issues related to the school inclusion of students with disabilities and at the same time a significant improvement of the services provided to students in general.

The applicative developed for this thesis is not for the exclusive use of students with disabilities, but they can be accessible to all students who attend multimedia classes in the university because these methods are useful to the improvement of teaching and learning in general.

1.2. Motivations

Nowadays most part of university lectures are carried out by professors with the support of Power Point slides, this approach is probably widely diffuse because it offers advantages for bringing considerable flexibility and resources to present multimedia content to students. Moreover it helps guiding the pace of a lecture and suppresses the needs for professors of explanations followed by handwritten content as it is in previous techniques. However, at the same time, it brings challenges for teaching and engaging the attention of students along the lecture.

Due the lack of interactivity, often high quantity of information presented at the same time followed by not so uncommon shallow explanations and the feeling frequently assumed by those who follows the lecture that by having later the digital content of the presentation gives a freedom that suppress the need to perform annotations; considerable number of students don’t follow properly the lecture, those who are distracted miss information being presented and consequently the quality of their notes are compromised.

These aspects may impact significantly the reduction of learning inside a classroom and even the study after class when students have in hands the copy of presentation but usually accompanied with poor or no notes taken.

Besides these issues; it is remarkable nowadays the presence of students following lectures using laptops claiming that it supports their annotations and the gathering of external content to support their learning. Although for a portion of students this principle works as stated, for others it is just a trap and fosters even more a distractive environment where a student can easily switch his attention from the context of the lecture to any other he feels more motivated at that time. The lack of software solutions that provides an appropriate support for the annotations is also a factor that contributes for this problem to happen.

The need for solutions that covers these problems just highlighted in a classroom based on multimedia content is what drives the goals of this project. The existent demand of software functionalities which incorporates principles of multimedia learning, the mechanics to support
the annotation process and resources that can actively engage a student on the content presented in class is the motivation of this work.

1.3. Objectives

This main objective of this work is to design and develop a software solution to support students during the annotation process in a typical university lecture in classroom, which is commonly based on presentation of multimedia content from slides followed by explanations of a professor.

The design of this software is aimed to be driven by multimedia learning principles in order to offer mechanisms to enhance the learning from lectures and adequate resources to cover the basic needs for the annotation process.

The second goal is through this software engage students to actively follow the pace of a lecture by reducing the workload to reproduce the multimedia content presented and suppressing the presence of distractive elements on a computer environment by offering an immersive and interactive Natural User Interface aiming that students reach another level of user experience that can indeed attract students attention to the content target of a classroom.

Therefore the design and implementation should consider aspects of recent technological outcomes in terms of hardware and software platforms to support the development of a Natural User Interface aimed. The software environment should also concern about integration with Polinotes notes package, which is a software solution that contributes to this project with functionalities for accessing the multimedia content being presented in a lecture.

1.4. Thesis organization

Chapter 2: Multimedia learning in classroom. This chapter describes the most relevant aspects associated with learning from a multimedia content in classroom.

Chapter 3: Process of annotation. This chapter discusses methodologies students use to take notes in classes, extract their main advantages and disadvantages in order to identify the most relevant requirements to be considered for the design.

Chapter 4: Natural User Interfaces: This chapter defines Natural User Interfaces and investigates aspects in this paradigm to be incorporated to the applicative in order to facilitate the process of annotation and improve multimedia learning.
Chapter 5: Designing for the applicative. It describes the most important design elements and concepts of an applicative on Windows Store environment and associates these elements with previously discussed requirements for proposing the design of the applicative target of this work to be implemented.

Chapter 6: Polinotes Student App implementation. Introduces Polinotes package as a solution for bringing the multimedia content from slides being presented and continues describing the implementation of design elements of Windows Store apps and the requirements for the process of annotation and multimedia learning.

Chapter 7: Conclusions. This chapter presents conclusions about this work and possible future directions to be followed from it.
2. Multimedia learning in classroom

This chapter presents the main aspects of a multimedia learning in classroom environment and proceeds exploring the most relevant cognitive processes students are involved when dealing with multimedia content. Later a discussion about the cognitive aspects for a software solution on multimedia learning is developed.

2.1. Multimedia learning

Multimedia learning is the term commonly used to define the cognitive theory that is concerned with studying the characteristics of learning content, which are presented in different forms, or using different types of media.

A multimedia message is based on a communication containing words and pictures intended to foster learning to students. The content of this communication can be delivered using any medium, including paper (i.e., book based communications) or computers (i.e., computer-based communications). Words can include printed words (such as you are now reading) or spoken words (such as in a narration); pictures can include static graphics — such as illustrations or photos — or dynamic graphics — such as animation or video clips. This definition is broad enough to include textbook chapters, online lessons containing animation and narration, and interactive simulation games [14].

According to [14], the learning from multimedia content can be measured by tests of retention (i.e., remembering the presented information) and transfer (i.e., being able to use the information to solve new problems).

The learning target related with the goal of this work is the one based on transfer and since we are dealing with multimedia content; we are particularly interested in the cognitive processes by which people construct meaningful learning outcomes from words and pictures.
2.2. Learning in classroom environment

From the definition established on [1], the learning process in classroom environment is affected by three components: the professor, the student and the design of the multimedia content presented on the lesson. Besides that, the environment presents specific characteristics and influences the communicative situation.

The student is characterized by its own preferences, habits, interest he has for that particular content and their own cognitive processes, measurable with the results obtained in previous courses.

The teacher presents the lesson that is characterized by its own style of teaching and exposure that varies from teacher to teacher and is therefore not analyzed in empirical terms.

The environment, finally, varies depending on the characteristics of the single discipline that is taught and depending on the time steps, these, for example, define the time that elapses between a lesson and the examination of the course.

![Figure 1 - Components of learning](image)

2.3. Methods for presenting content in classroom

In many classrooms the blackboard and whiteboards are still available for use by the teacher. The presentation of content in this modality is based on handwritten explanations. The handwritten can be considered ideal due the pace of presentation being appropriate for both
listening and for note-taking. Of course this advantage can fade out in case the teacher does not follow some basic rules (illegible handwriting, talking to the blackboard, shading the current writing).

However, it offers some problems since the writing remains visible to students until the teacher runs out of place and needs to erase something. Students are forced to take notes as long as they want to benefit from the lecture. Besides that, when revising the knowledge the students can go back to their own notes but not to the original content, in a huge lecture hall the board handwriting can hardly be visible from any place of auditorium, and handling this presentation type is very demanding for teacher.

Another methodology applied is the use of projectors and transparencies; most of the teachers did not continue to use on-the-spot handwriting presentation style, although it was still possible. Instead they switched to presenting the slides prepared in advance, either printed or handwritten. There was still the possibility to annotate the slides during the lecture, although not all the teachers took the advantage of this.

With traditional laptop and data projector replacing the overhead projector by means of electronic projection the comfort of spontaneous handwritten annotation was lost. The only way of attracting students’ attention was the laser pointer which is hardly visible in huge lecture halls [5].

Electronic projection is nowadays the most diffuse method, it appears often in the form of showing Power Point slides and enables professors to easily provide slides with diverse multimedia content such as graphics, and text information highlighted in colors and distributed in lists and videos [26]. Also students can follow the classes by performing annotations over printed versions of the slides being presented or having an electronic copy of the presentation document to support their studies afterwards.

According to [26], there are two main disadvantages with the use of e-projection. First, without proper training or experience, an instructor can encounter technical difficulties with the equipment. The second disadvantage is a critical problem for learning, which is the loss of interactivity. In such situation students don’t interact very much with the multimedia content offered, and consequently might create a feeling of dominating the high amount of information, which mistakenly believe that they fully understand, especially if the professor just reads the contents of the slide.

According to [1], one of the most significant negative points for multimedia learning based on slides is that students who attend a lesson often turns to be in a mode called "entertain mode", similar to the way people are when watching television. To support this behavior contributes a
passive environment, in which the lights are lowered to facilitate the slideshow and the contents are presented by the teacher without pause for students reasoning.

2.4. The multimedia content

Considering from nowadays situation with the widespread of multimedia content being presented through slides along the lectures, it is possible to identify some common patterns of content over slides and also derive some expected behaviors from professors while handling a presentation for each pattern.

The following list contains the most relevant scenarios identified; they are based on what this work assumes as the most common content dealt in multimedia lectures currently; and these scenarios are the basis of further analysis over multimedia content and solutions proposed for this project.

- **Slides in topics and subtopics**: It is commonly used to present key points of a lecture and offers an overview about the content to the students. Professors often add extra verbal explanation about the topics along this type of slide.

- **Slides with long texts**: Even considering the limitation of slides to have huge amount of texts, it is not difficult to find long texts on presentations. In this case the professor commonly repeats what is being presented and doesn’t add too much extra content through the explanation.

- **Slides with images**: The slide is composed by images and they are usually followed by additional texts contextualizing them. The purpose of each image is explained by professor.

- **Slides with graphics**: It is composed by graphics and usually followed by additional texts contextualizing the graphic. The professor in this case contributes analyzing the graphic, and explaining relevant concepts behind it.

- **Slides with formulas**: This slide contains formulas and commonly is followed by additional texts contextualizing them. The professor contributes explaining the formulas, emphasizes the concepts behind it and if relevant performs some demonstrations about the formulas using handwriting on whiteboard/blackboard or over projection in transparencies.
2.5. Cognitive processes for multimedia learning

This work proposes as starting point for the analysis of cognitive processes for multimedia learning dividing the responsible events into two components: internal and external.

External cognition is concerned with explaining the cognitive process involved when students interact with different external representations.

In the other hand, internal cognition is an analysis focused on cognitive processes happening internally from the moment the student is exposed to multimedia content. For this analysis, the Cognitive Theory of Multimedia Learning (CTML) of Mayer [14] can be considered more useful for this work since it provides directly empirical guidelines that may support a design of multimedia instruction more effectively.

In order to simplify the analysis, only the learning involved when a student receives information from the multimedia content presented by a professor in classroom environment will be discussed. Further cognitive process from interactions between students and student-professor in this same classroom will not be considered relevant for this work.

2.5.1. External cognition

External cognition is a phrase referring to ways that people augment their normal cognitive processes with external aids, such as external writings, visualizations, and work spaces. A main goal of external cognition is to explicate the cognitive benefits of using different representations for different cognitive activities and the processes involved. The main ones that can be identified in a classroom and is relevant for this work:

- **Externalizing to reduce memory load**: it comprises strategies for transforming knowledge into external representations to reduce memory load. One common strategy is externalizing things difficult to remember such as formulas, dates, complex sequence of procedures etc.;

- **Computational offloading**: occurs when a student use a tool or device in conjunction with an external representation in order to carry out some computation, for example by using a traditional pen and paper to solve a math problem;

- **Annotating and cognitive tracing**: it is related with externalizing the cognition by modifying representations to reflect changes that are taking place that the student
wishes to mark. Annotating involves modifying external representations; for example when a student follows a class using printed slides and change or add items related with some new explanation in class. In the other hand, cognitive tracing involves externally manipulating item in order to outline some element, for example when a student decides to underline or circulate some content based on its importance [10].

The components identified for supporting the external cognition are related with the annotation process. Students nowadays perform the tasks involved with these components by using since of sheets of paper as well as interacting with computer based solutions. A deeper analysis into annotation process is taken over the next chapter.

### 2.5.2. Cognitive Theory of Multimedia Learning

The cognitive theory of multimedia learning (CTML) is based on three cognitive science principles of learning; the human information processing system includes dual channels for visual/pictorial and auditory/verbal processing (i.e., dual-channels assumption); each channel has limited capacity for processing (i.e., limited capacity assumption); and active learning entails carrying out a coordinated set of cognitive processes during learning (i.e., active processing assumption).

- **Dual-channel assumption:** it is prevenient from Pavio’s research[13], and states that humans possess separate information processing channels for visually represented material and auditory represented material. This assumption is incorporated into the cognitive theory of multimedia learning by proposing that the human information-processing system contains an auditory/verbal channel and a visual/pictorial channel. When information is presented to the eyes (such as illustrations, animations, video, or on-screen text), humans begin by processing that information in the visual channel; when information is presented to the ears (such as narration or nonverbal sounds), humans begin by processing that information in the auditory channel.

- **Limited capacity assumption:** it claims that humans are limited in the amount of information that can be processed in each channel at one time. The constraints on our processing capacity force us to make decisions about which pieces of incoming information to pay attention to, the degree to which we should build connections among the selected pieces of information, and the degree to which we should build connections between selected pieces of information and our existing knowledge.
**Active processing:** it states that humans actively engage in cognitive processing in order to construct a coherent mental representation of their experiences. These active cognitive processes include paying attention, organizing incoming information, and integrating incoming information with other knowledge.

![Figure 2 - Cognitive Theory of Multimedia Learning](image)

The figure 2 shows a representation of the theory. Pictures and words come in from the outside world as a multimedia presentation (indicated at the left side of the figure) and enter sensory memory through the eyes and ears (indicated in the sensory memory box).

The boxes shown on Figure 2 represent memory stores, including sensory memory, working memory, and long-term memory.

- **Sensory memory** allows pictures and printed text to be held as exact visual images for a very brief time period in a visual sensory memory (at the top) and for spoken words and other sounds to be held as exact auditory images for a very brief time period in an auditory sensory memory (at the bottom). The arrow from pictures to eyes corresponds to a picture being registered in the eyes, the arrow from words to ears corresponds to spoken text being registered in the ears, and the arrow from words to eyes corresponds to printed text being registered in the eyes.

- **Working memory** is used for temporally holding and manipulating knowledge in active consciousness. The left side of working memory represents the raw material that comes into working memory — visual images of pictures and sound images of words. The right side of working memory represents the knowledge constructed in working memory, called pictorial and verbal models by the researcher and also links between them.
- **Long term memory** holds large amounts of knowledge over long periods of time and its content is transferred to working memory when a person needs to think about it (as indicated by the arrow from long-term memory to working memory).

The arrows among the memory stores presented on the Figure 2 are the cognitive process identified by Mayer as the responsible for multimedia learning.

- **Selecting relevant words**: it involves paying attention to some of the words that are presented in the multimedia message. If the words are presented as speech, this process begins in the auditory channel. However, if the words are presented as on-screen text or printed text, this process begins in the visual channel and later may move to the auditory channel if the learner mentally articulates the printed words. The need for selecting only part of the presented message occurs because of capacity limitations in each channel of the cognitive system.

- **Selecting relevant images**: it involves paying attention to part of the animation or illustrations presented in the multimedia message. This process begins in the visual channel, but it is possible to convert part of it to the auditory channel (e.g., by mentally narrating an ongoing animation). It is not possible to process all parts of a complex illustration or animation so learners must focus on only part of the incoming pictorial material. Finally, the selection process for images — like the selection process for words — is not arbitrary because the learner must judge which images are most relevant for making sense out of the multimedia presentation.

- **Organizing words**: in this process the learner builds connections among pieces of verbal knowledge. This process is most likely to occur in the auditory channel and is subject to the same capacity limitations that affect the selection process. Learners do not have unlimited capacity to build all possible connections so they must focus on building a simple structure.

- **Organizing images**: In this process, the learner builds connections among pieces of pictorial knowledge. This process occurs in the visual channel, which is subject to the same capacity limitations that affect the selection process. Learners lack the capacity to build all possible connections among images in their working memory, but rather must focus on building a simple set of connections.
• **Integrating**: It involves building connections between corresponding portions of the pictorial and verbal models as well as knowledge from long-term memory. This process occurs in visual and verbal working memory, and involves the coordination between them. This is an extremely demanding process that requires the efficient use of cognitive capacity. The process reflects the epitome of sense making because the learner must focus on the underlying structure of the visual and verbal representations. The learner can use prior knowledge to help coordinate the integration process.

### 2.6. Learning principles

From his research, Mayer defines principles to be followed in order to promote learning for students from a multimedia content. These principles are based on how cognitive science understands the limitations of working memory and methods for encoding into long-term memory. Follow the enlisting of his principles:

- **Multimedia principle**: Students learn better from words and pictures than from words alone.

- **Spatial contiguity principle**: Students learn better when corresponding words and pictures are presented near, rather than far from, each other on the page or screen.

- **Temporal contiguity principle**: Students learn better when corresponding words and pictures are presented simultaneously rather than successively.

- **Coherence principle**: Students learn better when extraneous words, pictures, and sounds are excluded.

- **Modality principle**: Students learn better from animation and narration than from animation and onscreen text.

- **Redundancy principle**: Students learn better from animation and narration than from animation, narration, and on-screen text.

- **Individual differences principle**: Design effects are stronger for low-knowledge learners than for high-knowledge learners and for high-spatial learners than for low-spatial learners.
2.7. Conclusions for a software solution to support multimedia learning

According to [10], a general cognitive principle for interaction design based on the external cognition approach is to provide external representations at an interface that reduce memory load and computational offloading. In this way it can extend or amplify cognition, allowing users to perceive and do activities that they couldn’t do otherwise.

Using the context of multimedia learning, the extension of cognition provided by computer based tools that offer any computational offloading or reduce memory load are beneficial since the student has to apply less cognitive effort in order to perform tasks such as doing a calculation on a piece of paper or even annotating for retrieving key points of a topic when reviewing his notes; and therefore using more of their cognitive capacity in order deal with more content and consequently learning more.

An important point for a software solution when reflecting about external cognition is to present an interface offering low levels of cognitive effort to be learned and mastered; therefore the student doesn’t have to apply many cognitive resources to learn something else besides the multimedia content already in place.

The interface should also concern about not showing content or suppressing interactive elements different from the target multimedia content of a lesson in order to improve the learning, as it claims the coherence principle of Mayer; as well as it should be focused on covering the main tasks existent on the process of annotation in order to increase the capacity of the external cognition.

Analyzing Mayer’s conclusions over multimedia principle, spatial principle and temporal principle; a software solution to support multimedia learning should bring together words and pictures associated with a certain topic. These elements have to be brought at the same page and near each other. They also should be presented at the same time in order to foster learning from multimedia content.

It works due Mayer proposition that the channels of working memory process auditory and visual information separately. Thus a learner can use more cognitive processing capacities to study materials that combine auditory verbal (words from speech or written text converted to the auditory channel) information with visual graphical information. In other words, the multimodal materials reduce the cognitive load imposed on working memory. When an illustration or animation is presented, the learner is able to hold only a few images in working memory at any one time, reflecting portions of the presented material rather than an exact copy of the presented material.
Considering the aspects discussed about the “passiveness” of a student when exposed to a slide based lecture, a software solution should support annotations and create mechanisms to enhance the user experience in a way to incentivize students to manipulate the multimedia content could be valuable in order to suppress the “entertain mode”.

Another issue is related with presenting multimedia content is the speed with which the concepts of a lesson are exposed; in classes where slides are passed fast followed by shallow explanation doesn’t allow students to settle the information and actively think about the content by themselves.

Regarding this aspect, a software to support students for multimedia learning should be able to browse easily and quickly along multimedia content and allow students to modify and add personal content to notes previously taken without demanding much cognitive effort interactions with the interface; these features can make students to return to some state of the lesson that was inappropriate fast presented, engage them to think actively about that content presented and therefore create the mental connections that are the basis of a rooted and long term knowledge.

Moreover due the high presence of elements on slides presentations in which the readability and comprehension decreases as the distance from the projection such as texts, formulas and graphics; a software solution should considerer this aspect and bring these elements to the students in order to facilitate the readability of the content for those far from the projection or have some vision impairment that could compromise the comprehension of material due the distance.
3. Process of annotation

This chapter describes the process of annotation and methodologies students apply in order to perform it. It proceeds by identifying and comparing the most relevant tasks and elements carried out among the diverse methods found. Later the conclusions for a software solution aiming to support the annotation process are then performed.

3.1. The process of annotation

Note taking occurs in frequent and various everyday life situations. To make purchases, to plan future events and activities, to study for examinations, to prepare a technical talk, to design a model in an industry, and to record the minutes of work meetings are a few examples. Furthermore, the reasons why individuals take notes are highly variable. Despite the diversity of contexts and intentions, all notes taking entails recording information collected from one or several sources. Such a record constitutes a stable external memory that is intended to help to plan a future activity, to learn, to think, or to create. It is important to understand this common activity for both theoretical and practical reasons [3].

Note-taking is the practice of recording information captured from another source. By taking notes, the writer records the essence of the information, freeing their mind from having to recall everything [2].

Notes can be defined as short condensations of a source material that are generated by writing them down while simultaneously listening, studying, or observing. Their function is to gather information distributed in a lecture, a book or in any other situation that needs to be remembered. In other words, notes are external memories whose content is more or less explicit [3].

From a cognitive point of view, according to [19], involves a range of underlying mental processes and their interactions with other cognitive functions. Note-takers need to acquire and filter the incoming sources, organize and restructure existing knowledge structures and, most importantly, they must store and integrate the freshly processed material. The process of note-taking depends largely on the ‘working memory’. When taking notes of a presentation, we
maintain a short-term memory buffer in order to acquire, mentally represent, select and understand the continuous flow of incoming new information and to update and interact with the already-stored knowledge. Working memory during note-taking contributes to processes such as cognitive load, comprehension or writing.

However, as discussed on [19], since note-taking is constrained by the capacity of the working memory, students may choose between two main strategies to suppress this issue along a lecture, with this choice being susceptible to change while taking notes. They may reduce their activity either to comprehension (listening or reading and noting the less possible) or to transcription (without processing the content of what is heard or read in order to be able to transcribe the maximum of information).

In the second case, a common way to note takers deal with it is by giving up the transcription of letters, words, part of sentences, even of whole sentences, and therefore using abbreviating procedures. Nevertheless, the use of such procedures is not always effective for resolving the variation of rate between fast oral production and slow writing [19].

### 3.2. Annotation for multimedia learning

According to [18], educational psychologists report that note-taking has two main functions: encoding and storage.

- **Encoding function**: it is essentially the process of writing of notes. The annotation of information facilitates the learning even if the student doesn’t review the notes later. This is usually explained due the process of taking notes, the student has to re-express those some inputs from the instructor or multimedia content, and while doing so, the ideas get mentally rehearsed, integrated at a deeper level, or re-encoded mentally in a form that is easier to and remember

- **Storage function**: it is about reviewing the annotation taken before in order to facilitate the memorization of content. This function allows students to consolidate the information recorded, to reconstruct concepts not recorded accurately and to learn the information again forgotten, by recalling the key concepts of a lesson. The improvements made by the storage function are more evident the more we go forward in time.
The effectiveness of these functions over multimedia learning, according to [1], is strictly associated with the quality and completeness of the notes compiled. The more notes are complete and structured, more students will be able to recall the concepts written after a careful study. Another aspect is the learning from the storage function is increased when the student who studies by the notes is the same as the one who wrote the notes.

And still according to [1], there is a strong relationship between the student identifying the concepts professor thinks are important and which are negligible within a lesson and the development of more complete and effective notes from the multimedia content, which as mentioned before contributes with a better learning.

Moreover, according to [3], students have to deal with several problems related to the flow of information when writing notes from a speech of the lectures. The note taker is constrained by the rate of speech of the lecturer: writing speed is much slower than speaking speed. In other words, note takers must maintain an active representation of what they are hearing in order to get sufficient time to exploit and to transcribe a portion, while being faced with a continuous updating of the message content as it is spoken.

### 3.3. Properties of class notes

The research of [18] pointed out some common properties among notes found on lectures. These properties were based mainly on observations of the researcher over a small corpus of classes notes gathered from engineering students at the University of Tokyo and in the following are described the most relevant for this work.

- The most typical use of class notes is for review by the person who created them. As such, notes do not have to be stand-alone documents; rather they are good enough if they serve to help retrieve memories of what the instructor was saying or writing.

- Class notes are used typically reviewed during the semester; they are not used as permanent records. For these reasons class note are generally rather “sloppy”, in terms of layout etc., compared to other documents.

- Text generally is in short chunks: full sentences are rare and paragraphs non-existent. Most instructors write only key phrases on the board, explaining the relationships between phrases only orally and/or graphically, and most students follow suit.
• In contrast to the essentially linear nature of many documents, class notes are essentially two-dimensional, reflecting most usually instructors’ use of blackboard space.

• Text and graphics often appear together. For example, text is underlined, words are connected with arrows, and phrases appear in boxes, and so on.

• Class notes can be moderately long. A student in a lecture generally sets out to absorb everything, and typically records as much as possible of the new class content.

• Class notes are usually written under time pressure.

• Most classroom note-taking is done sitting down, at a desk with enough space to rest a pad of paper, or a notebook computer and maybe a mouse.

• Class notes are revised relatively infrequently. In contrast the time involved in creating many documents of other types, such as articles or plans, is typically dominated by the reorganizing, rewriting, and other editing phases.

3.4. Methods for annotation in classes

This section presents results of a quantitative study [6] carried out to investigate the methodologies chosen by students while performing annotations and the reasons pointed out by them to justify their choices to produce notes. This study approached 408 students under four different university courses belonging to different faculties (316 students from Computer Science and 92 students from Pedagogy) at the University of Darmstadt.

The methodology to obtain data about student’s preferences was based on a questionnaire with several questions both open and closed regarding aspects of the process of annotation.

From the answers given during the questionnaires, it can be identified that not all students take notes during classes and the percentage varies depending on the discipline; while 93.3% of the students of pedagogy keeps writing the contents displayed by the teacher, only 62.3% of students in computer science courses take notes during class. Those who don’t take notes justified by considering the slides provided by the instructor contain sufficient information; students also indicated that taking notes distracted them from listening the professor explanation.
For those who take notes, the methods chosen are mainly the following:

1. Simple white sheets,
2. The paper version of the slides of the course
3. The printed version of the lecture notes of the course
4. Laptop

The Figure 3 shows the percentage of note takers on single media or in combinations of several media. Both in computer science and pedagogy, traditional note taking with a pen and paper clearly outperforms notes on a laptop. In the computer science courses, 77% of the students took their notes exclusively on paper. This group consists of three subgroups of roughly equal percentages which took notes either on empty sheets of paper, on printed course slides or on both of them. 8% made an exclusive use of a laptop, while 15% indicated to prefer cross-media note taking, which combines notes on a laptop with notes on empty sheets of paper or on printed course slides.

![Figure 3 - Methodologies for taking notes](image)

The context factor of the discipline proved to be an influential factor of the context model, since laptop use differed largely between the disciplines. In the pedagogy course, laptop use was almost not existent. 98 % took their notes exclusively on paper. The two largest groups (about
45% each) took notes either only on empty sheets of paper or combined them with printed slides or the printed course script.

The questionnaire also asked the students to indicate the benefits that led them to prefer a solution over another. They were asked to evaluate on a scale from one to five the importance of some of the advantages of using a pen and paper and a laptop. Besides the numerical evaluation, was also required a written explanation to each student to know the main reason about annotation support.

The graph on Figure 4 highlights some of most relevant finds. From this research it can be concluded that students choose to use pen and paper mainly because this type of support ensures greater flexibility of annotation and the ability to perform free drawn on the sheets. This factor is followed by the fact that the sheets are easy to transport and does not constitute an impediment logistics for students.

On the other hand written records by electronic means, such as laptop, ensure ease of long-term storage, quality and readability of the notes more than the paper. The use of notebook and laptop also facilitates the students in the search for keywords within the text and make notes easily shared among students.

![Figure 4 - Advantages of paper and electronic notes](image-url)
3.4.1. Pen and paper

According to the research previously mentioned, students still widely use the traditional pen and paper as tools for note taking during classes. Among those, they are equally divided into those who use empty sheets and those who write notes on the paper version of the slide, or on both substrates.

When using empty pages the student has to do a considerable cognitive effort to reproduce part of the content presented, select the information and synthetize into his own notes. In other hand, since it is based on free space it is easier to structure the annotation but inconvenient in terms of time consuming and quality when the student has to copy long texts or perform complex draws in order to reproduce something valuable for the meaning of the notes.

Annotations on printed course slides are regarded as advantageous, since these allow establishing a direct reference to the context by taking the note on the place they refer to, due part of content is already in place, the students can have more time to focus on writing down extra information provided by professor or even their own synthesis of content. A recurrent problem in this approach is related to structure the annotations, usually along long written texts some notes overlap previous content and also issues with the lack of space to write properly are found.

3.4.2. Computer based annotation

In contrast to document creation, which has long been one of the classic "killer applications" for personal computers, the creation of notes, that is, of records primarily intended for ephemeral or personal use, has only recently become a candidate for computer support. However, thanks to the recent proliferation of digitizing tablets integrated into various mobile devices, paper is becoming obsolete in more and more domains [18].

As the use of the software during the process of annotation has not yet been studied in detail by the community scientific, this section aims to bring diverse solutions available for students to help the process of annotation. For this scenario, it was considered recent technologies for performing annotations in terms of software and hardware.

In terms of hardware, as shown by [6], laptops are still the main resource used by students, however with recent advances on multi-touch devices and consequently reduction of prices for this technology it is becoming common the usage of Tablets and Tablet-PC for performing annotations.
Analyzing the software perspective, according to current software solutions found, this work identifies two main set of solutions aiming to support the annotations of students: reproducing content through document editors and annotating over document visualizers.

- **Document editors**: offers powerful tools to add and format content, usually when a student use this software he reproduces all the content presented in class and adds in sequence his own perceptions to the notes.

- **Document visualizers**: essentially opens files containing the content being presented in class (usually supports Power Point and PDFs formats) and also offers resources to students to add his own notes over the content.

In sequence it is presented some of the most relevant softwares found for annotations purpose, as well as their main functionalities and design approach to help students in the task of notes taking.

**Microsoft One Note 2010**

This is one of the most complete document editors supporting annotations in the market; it supports inserting images, audio and video-recording, screen clipping and drawings. It also supports importing from web pages or any Windows application by means of a virtual printer. It is possible to attach files of any types to the note, similar to email attachments. Can search text, audio, pictures and handwritten input in Microsoft Tablet PCs, it disposes of a system of Optical Character Recognition (OCR), which is able to identify the characters written by using a pen and translate them into the digital character. Files can by synchronize to multiple desktop PCs, laptops, mobile phone, a Windows Live SkyDrive account or to a SharePoint site.

According to the conclusions taken from [1] about performing annotations with this software, it presents a not pleasant user experience when user wants to make use of multi-touch capabilities. The touch of the fingers can be interpreted in two different ways depending on which mode is One Note at that moment: selection mode or landscape mode. While the first allows users to move and resize the objects in the digital sheet notes, the second handles the navigation throughout the document notes. The mode change is not automatically recognized by the software and requires the user to press a button often in the GUI software, which makes the user experience less linear and less intuitive. This issue can make the user annoyed and offer additional effort while interacting and therefore slowing down the process of annotation.
It is possible to find similar softwares running for different platforms such Circus Ponies Notebook (works on Mac environment) and Evernote (runs in diverse platforms, including event mobile).

![Image of Microsoft One Note](image_url)

**Figure 5 - Microsoft One Note**

**LiveNotes**

The main goal of this project is to offer features for sharing notes taken among a small group of students, it is based on the statement that fostering the dialogue and discussion among students makes easier the learning process. It makes use of a wireless network, allowing students to cooperate in order to produce annotations of a lecture within a same document, and is designed to be used on a Tablet PC.

Topics presented in a lecture can be used to enrich the notes by importing inside Microsoft Power Point slides. It offers the possibility to members of a group to interact with the content
presented by professor and confront it, making directly references to the slides on the notes document.

Considering the design perspective, it also makes use of the paper notebook metaphor, with the whiteboard interface implies the idea of the page that the user is currently viewing and allowed to edit. The navigation through pages is designed to be done through a widget at the top of the main window. This widget indicates the current page number of each user is using an oval icon displayed according to the user's selected ink color and represents along the slider the position relative of all other users. The user moves between pages by dragging his icon along the slider. To allow for flexibility, automatic page turns are not supported and the user is responsible for advancing to the next slide. The whiteboard provides a public view of the local user's current page, such that every user's writing or keyboard input on that page can be seen by other members in the same group.

A problem regarding this solution is considering the space management, in which users may write on top of another user's notes, it can be considered a big issue if there are a considerable number of students interacting in the same time with same content. However, in order to suppress this issue, slides imported into the whiteboard's background occupy only two-thirds of the top left-hand whiteboard area and therefore offering the remaining space to the users perform their annotations [29].

![Figure 6 - LiveNotes](image-url)
NotePals

In this system users can make notes through a simple hardware such as Personal Digital Assistant (PDA) using ink-based input. The software provides functionalities for storing the user notes in a central repository, privacy control by defining which notes are private or public according to assigned groups and sharing the personal notes among users. Besides that, it also provides a secondary way to interact by providing access through a web browser for visualization of the content store.

In principle, considering the device used (a PalmPilot based interface), it offers big challenges for designing due the small size screen which creates obstacles to draw on considering the two-inch square screen, user’s hand can obstruct easily the process of annotation and the visualization and small resolution would allow only small words to be written so making it even hard to read.

The designed adopted managed to deal with these problems and it is shown in Figure 7(a). It has an area called “Overview Area”, at the top of the screen, where users can write in their own handwriting directly on the page and also the “Focus Area”, in the box at the bottom of the screen where words appear increased by 40% comparing with Overview Area original size. Once the user has filled up the focus area with text, a quick right to left swipe of the pen moves the focus cursor forward, clearing space for the next word. This interface allows many words to fit on one page, despite the small screen.

Each page of notes in NotePals is created within a “project.” Projects are organized in a hierarchical set of folders, as in Figure 7(b), which gives users a way to group notes into topics. To give extra context, users can assign each page of notes a “Type” that indicates what kind of information it contains (e.g., action item or new meeting header) [31].

![Figure 7 - NotePals interface](image)
StuPad

This software was developed to be used on a Tablet-PC and essentially enables students to take individual notes on top of background slides using tablets. It offers visualization of two channels of information: the one generated by the professor where is replicated as an image of the professor slides and another where the student can write his personal notes. Also it is available the list of websites visited by the professor during the lecture. It is meant to be used during annotation process and later for revision and study of the notes taken before.

Regarding interaction design perspective, the software makes use of an affordance by representing the traditional pen and paper interface to make notes and also incorporates the flip gesture to turn the pages. As shown in the Figure 8 the students have access easily to the content presented along the lectures through a Navigation Bar where it is provided additional information that aims to help students to track the content offered by professors with their personal notes [32].

Figure 8 - StuPad interface
Windows Reader

This software comes as one of the standard document visualizer on Windows 8 Store environment. Its purpose is to offer an interface for document reading, mainly for PDFs documents. It is possible to interact by only using multi-touch inputs for accessing documents, reading, navigating through pages and zoom in/out functionalities.

Considering the pen-based input support, it can be used also as software with annotation purpose by adding the handwriting along the pages of a document. The user can save and open later the document with his notes appended; therefore offering a powerful tool for supporting this task.

The interface provides a very small set of functionalities but covers very well the essential operations to carry out the task to read and perform simple annotations along documents. Considering the fact it is completely integrated with Windows 8, it provides to the user a pleasant user experience when interacting with documents.

Figure 9 - Windows Reader interface
One Note App

This applicative also runs as Windows 8 Store App; it can be considered a light version of the One Note already placed in the Microsoft Office environment and contains all the basic resources to support annotations but with functionalities optimized to be executed in multi-touch and pen devices.

It is noteworthy the design multi-touch of operations such as to select objects (user can contour all the area he wants to select using touch) and move elements along the documents without changing states (differently from the One Note 2010 as already mentioned). Also selection of modes for text edition with radial interfaces provides a set of tools very well appropriated for interactions using fingers.

![Figure 10 - One Note App Interface](image)

3.5. The tasks of annotation process in classes

In order to have a better comprehension of the process of annotations, this section identifies and confronts the main set of tasks and objects managed by a student (physically or virtually) among different methodologies of taking notes. First it is considered a scenario using pen and paper and then later a scenario of an annotation with a support of softwares.

The tasks described are related with the simplest action to be done in order to change the state of an element present within the annotation content.
3.5.1. Pen and paper

Considering initially the use of pen and paper, the main elements or objects this work identified that are present on annotations using this approach are handwritten, illustrations and the page itself.

- **Handwritten**: the result of a student writing something on a paper, it can be a sequence of words composing texts or numbers representing a sequence of calculations or formulas;

- **Illustrations**: additional elements that can aggregate some meaning to the notes, it is draw by the student and can be representations such graphics or abstractions to add some semantic such as arrows and brackets;

- **Page**: it is where the result of annotation is placed, the student has to manipulate in order to place in a comfortable position to write down, read content or search for something.

As discussed before, two main scenarios can be identified for taking notes using pen and paper: when a student uses empty pages and other when he takes notes over printed slides related with the lecture.

The way printed slides is disposed along the pages may influence considerably the manner how students will perform the annotation. A layout which maximizes the quality, which is printing one slide per page and then using the rest of page for annotation in general is not used by students since it represents more costs for printing and also the student has to carry more pages with him along the lectures. It is common to find formats that can be derived as more economical and brings more flexibility to carry all the pages such as printing two or more slides in the same page.

Breaking these scenarios according to annotation elements identified and the set of tasks that can be performed by students over that elements, we can achieve a level where a qualitative study can be taken in order to compare and highlight the differences of those methods. This comparison is illustrated on the Table 1.

Considering annotation as a time constraint activity [18], since the dynamic of a class is about continuously presenting new content and the student has to be constantly taking notes, it would be reasonable to think that the quality of an annotation is associated with how much information a student can process from a lecture and represent it over a paper in a structured way.
Table 1 - Comparison between annotations over empty pages vs. pages with printed content

Assuming this principle, the advantage of using a methodology of annotation over printed content offers gain in time and allows the student to have more time to reason about the content he is dealing with, however as shown in the table the structure of an annotation can be compromised due lack of space to execute basic tasks.

As expected for being the simplest method, an empty page offers all the flexibility the student can have to structure his notes. On other hand the time constraint issue can influence in the quality since tasks to reproduce content, mainly graphics, are time consuming.

Another issue is regarding the page management, the student faces problems when using considerable amount of pages at same time on his desk. He also has to reoccur to other physical objects to store and organize in some order to facilitate retrieving process. This process takes time, the student can lose content being presented or due the lack of time not organize properly the way notes are indexed compromising his studies later.
3.5.2. Computer based annotations

Using the same approach and analyzing the diverse softwares previously presented for annotation purpose, it is possible to identify the common elements and actions to be performed over them. The most relevant elements found are:

- **Handwritten**: the result of a student writing something using an electronic pen;
- **Text**: represents the text typed using a traditional keyboard or a virtual keyboard on touch screen technology;
- **Images**: image elements that add some meaning to the notes, they can be photos taken, objects to add some semantic between texts such as arrows;
- **Page**: the metaphor of physical page created by these softwares, this element usually offers functionalities to move along other pages and to position itself in order to orientate the user where he wants to place his notes;

As mentioned before, the softwares used for annotation purpose can be divided between text editors and content visualizer. By performing the same analysis between these groups as the one brought by comparing empty pages and pages with printed content, the conclusions are at some point similar. For content visualizers the same problems of lack of space for structuring annotation and advantages in productivity by taking advantage of a previous content. And for text editors it has advantage on the flexibility of using a full page for structuring and developing the content but still some effort has to be taken in order to reproduce content presented in class.

In other perspective, by analyzing the way students interact with the computer solutions for taking notes presented, it is possible to identify two main groups of interaction modes that changes considerably the way actions are done, the first and more traditional is using keyboard and mouse and the other using multi-touch devices and electronic pen.

In order to comprehend how changing the way of interaction contributes for the annotation process, the Table 2 contains an analysis over how users perform actions on the basic elements identified in this section.

Considering the interaction using mouse and keyboard, it is clear that the accuracy offered by tracking position of cursor offers valuable resources when dealing with digital content. It is ideal for basic operations such as selecting certain position of texts, navigating along pages and even moving faster and precisely elements along the page.
<table>
<thead>
<tr>
<th>Element</th>
<th>Action</th>
<th>Keyboard and mouse</th>
<th>Multi-touch and pen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handwriting</td>
<td>Write a text</td>
<td>Not applicable</td>
<td>Pros: brings the handwriting to a computer system</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cons: demands some training for electronic pen</td>
</tr>
<tr>
<td>Erase a text</td>
<td></td>
<td>Not applicable</td>
<td>Pros: pen usually has eraser mode, faster and easier than erasing on physical paper</td>
</tr>
<tr>
<td>Move text</td>
<td></td>
<td>Not applicable</td>
<td>Pros: easy and natural to move to the desirable place</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cons: some software solutions demands changes of states/modes or selection before performing move</td>
</tr>
<tr>
<td>Text</td>
<td>Type a text</td>
<td>Pros: fast and widely diffuse to insert text using keyboard</td>
<td>Pros: easy to allocate text and insert in different positions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cons: need lights to see keyboard for typing</td>
<td>Cons: virtual keyboard to insert notes might not be practical for long texts</td>
</tr>
<tr>
<td></td>
<td>Edit a text</td>
<td>Pros: accuracy to select positions and part of texts to edit with mouse cursor</td>
<td>Pros: might be practical use pen to select fragment of texts to be changed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cons: need lights to see keyboard for typing</td>
<td>Cons: virtual keyboard to insert notes and touch to point position and select part of texts</td>
</tr>
<tr>
<td></td>
<td>Move a text</td>
<td>Pros: selection of text and operations to define new position are usually easy to perform</td>
<td>Pros: easy to move and natural to move with touch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cons: some software solutions demands changes of states or selection before performing move</td>
<td>Cons: some softwares demands changes of states or selection before performing move, also it can offers challenges to user when determining fragment of text to be moved</td>
</tr>
<tr>
<td></td>
<td>Erase a text</td>
<td>Pros: accuracy to select positions and part of texts to delete with mouse cursor</td>
<td>Pros: might be practical use pen to select fragment of texts to be removed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cons: when touch is used to select parts of a text to be deleted</td>
</tr>
<tr>
<td>Illustration</td>
<td>Add an illustration</td>
<td>Pros: user can append images easily</td>
<td>Pros: it is easy to draw with a pen and possibility to append complex images from computer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cons: difficult to draw something with mouse, it depends on software functionalities to help on the task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Move an illustration</td>
<td>Pros: it is accurate to use a mouse to move illustrations</td>
<td>Pros: easy and natural to move to the desirable place with touch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cons: some software demands changes of states/modes or selection before performing move</td>
<td>Cons: some software demands changes of states/modes or selection before performing move</td>
</tr>
<tr>
<td></td>
<td>Remove an illustration</td>
<td>Pros: pen usually has eraser mode, faster and easier than erasing on physical paper or using a mouse</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Next/previous page</td>
<td>Pros: easy to manipulate scroll bars, mouse wheel and keyboard help this task</td>
<td>Pros: natural to use fingers to move along different pages</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cons: not practical when it has big amount of pages</td>
</tr>
</tbody>
</table>
Design and implementation of a Natural User Interface to support the annotation process for multimedia learning

Diogo Borges Krobath

<table>
<thead>
<tr>
<th>Manipulate a page</th>
<th>Pros: provides good control of positioning and level of zooming of the page</th>
<th>Pros: gestures imitate the physical way of interacting for moving and zooming</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Cons: usually it needs change of states between page manipulation and content</td>
</tr>
</tbody>
</table>

**Table 2 - Comparison between annotation with keyboard and mouse vs. multi-touch and pen interactions**

Besides that, since years of training, users are adapted to keyboards therefore typing using it is considered normal and fast. This interaction paradigm is widely diffused along document editors thus offering an easy learning about how to deal with functionalities of the software.

However the introduction of handwriting input through electronic pens into a digital document is extremely welcomed for any software with taking notes purpose, the natural way of writing associated with computers resources can provide powerful tools.

The main disadvantage for multi-touch and pen approach is related with precision of selecting specific content (selecting fragment of texts for example) and also constantly changes of states of interfaces in order to perform actions (such as between moving content and editing content as the way Microsoft OneNote is designed).

Nevertheless, it can be found software that design the pen to be also able to be used as a cursor and consequently help to improve performance of tasks involving specific selection. At the same time, different design approaches could be taken in order minimize changes of states that are demanded often by the user.

### 3.6. Conclusions for improving process of annotation

Regarding the software question, the needs of students taking class notes are fairly clear, and they are not satisfied by existing drawing tools or editors. Important features require to meet these needs include the use of pen, keyboard, and mouse-or-equivalent, support for fast text entry at arbitrary positions, and support for common text decorations and drawing sequences. Barring the appearance of new technologies that radically change the nature of text entry, note-taking, or human learning, any application for taking class notes will probably need these features [18].

In respect to the interactivity, from all the analysis performed in this chapter it is possible to conclude that a software solution for supporting annotation should be designed to support pen input. It is reinforced by an experiment done by [18], in which students were asked to copy a
page of notes, confronted the usage of computer based annotations with a mouse versus pen and paper, concluded that it takes about twice as long, on average, with the computer than with pencil and paper to do the task. The most salient problem was the difficulty of drawing diagrams etc. with the mouse.

In certain situations a pen computer by itself is not enough for taking class notes. Although the presence of graphic elements is a hallmark of class notes, it is nevertheless the case that most class notes consist mostly of text.

Text input from keyboard is an advantage when a system of OCR recognition is not stable enough to carry all the diversity of student’s notes (by considering the diverse styles of handwriting, abbreviations and even different languages) or even for situations where the hardware doesn’t offer a reasonable digitizing resolution for handwriting input with a pen. Besides that typing produces text which is more editable, more searchable and easier to be read later. Also, since many users are already very familiar with this interaction, keyboard input is faster and less tiring than writing on paper.

Based on these arguments and considering a software solution independent of possible hardware limitations, a note-taking system should provide support for both pen and keyboard: a pen for drawing a variety of graphic elements and few handwritten words, and a keyboard for large text input.

Considering the time cost of keyboard-to-pen switching is greater than the time cost of keyboard-to-mouse or keyboard-to-trackball/ touchpad switching, note-taking systems should include a mouse or equivalent [18], the solution should also be able to be interactive with mouse modality input in a way to replicate the functionalities already place for pen, touch and keyboard but being handled also by the mouse.

Similarly, both pen and mouse should be available as pointing devices, for operations such as defining a fragment of text to be edited they are still much more appropriated that using touch modality.

Analyzing the basic cognitive functions identified by researchers for learning through annotations, a software solution for this purpose should be able to maximize the effectiveness of encoding and storage functions.

For encoding, all the resources of manipulating the objects of annotation should be provided. Besides the common functions to insert and edit content, the solutions should care about an appropriate design for resources aiming to structure the annotation and move objects for creating more space for new content. This is justified as previously mentioned the more an annotation is structured and organized the more it provides a better learning to a student.
Moreover, it can be concluded as extremely useful providing to students an easy access to the content presented in class in order to save some time from replicating and therefore applying that time in producing notes with better quality and more structured.

According to the affirmation that associates the ability of students of identifying key concepts of a lecture and the quality and completeness of notes, the software to support annotation should be able to identify key concepts and highlight it to support the user notes. From the diffuse use of Power Point slides in lectures nowadays, the key contents on a slide can be identified in different forms, such as titles, definitions and bullets.

In the other hand, for storage mode, the user should be able to browse and view easier his records on the document. Considering the multimedia content from Power Point slides, the document of annotation should be aligned with the power point slide its content came from in order to facilitate user to bridge his notes and the content mentioned in class.

Another useful aspect for storage function is related with retrieving annotations documents and the way they are organized; an efficient way to do it might reduce efforts and time for finding a specific document and consequently allow the student to continue executing his studies with less software interruptive contexts.
4. Natural user interfaces

In this chapter are discussed the main properties of Natural User Interfaces. It analyses the learning aspects while interacting with this modality of interfaces, presents the main features of the new technologic outcome Microsoft Windows 8 and later discusses the adoption of this paradigm as solution for this work.

4.1. The evolution of interfaces

While the increase in computing power had been more or less continuous, the interfaces between human and computers have evolved more discontinuously. A widely held perspective is that interfaces have passed through phases. These phases are loosely defined but can be thought of as phase of typing commands (command line interface), followed by graphical user interface (GUI) and nowadays Natural User Interfaces (NUI) are appearing [8].

A command-line interface (CLI) is a means of interaction with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines). The CLI was the primary means of human interaction with most early operating systems and the interface is usually implemented with a command line shell, which is a program that accepts commands as text input and converts commands to appropriate operating system functions.

A Graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices using images rather than text commands. GUIs can be used in computers, hand-held devices such as MP3 players, portable media players or gaming devices, household appliances, office, and industry equipment. A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. The actions are usually performed through direct manipulation of the graphical elements.

Most computers with which people interact nowadays are based on the desktop metaphor and rely on a known set of user interface elements, commonly referred to as WIMP (windows, icons, menus and pointers), which stands for:
● **Windows**: self-contained program, isolated from other programs that run at the same time in other windows;

● **Icons**: shortcut to an action the computer performs;

● **Menus**: text or icon-based selection systems that selects and executes programs or tasks;

● **Pointers**: onscreen symbol that represents movement of a physical device that the user controls to select icons and data elements;

A class of successful applications using GUI paradigm and relevant for this project is document processing. This class introduces new concepts such as “what you see is what you get” (WYSIWYG) that is, what is on the screen is a reasonably faithful rendition of what will be printed. It fundamentally shapes the way people spend time composing text – simultaneously focusing on both text and content. It implies a user interface that allows the user to view something very similar to the end result while the document is being created. In general WYSIWYG implies the ability to directly manipulate the layout of a document without having to type or remember names of layout commands [8].

Nowadays due the most recent technological outcomes, Natural User Interfaces (NUI) are emerging (its properties are discussed in the following subsection) and it seems to be in a position occupied by the GUI in the early 1980s. Like the desktop GUI, NUIs promise to reduce the barriers to computing still further, while simultaneously increasing the power of the user, and enabling computing to access still further niches of use. But just as GUIs did not simply make command systems easier, NUIs are not simply a natural veneer over a GUI. Instead, like GUIs, NUIs have a set of strengths based on what they make easier, bow they make those things easier, how they shape the user’s interaction with technology, which niches they fit in, and whether or not these niches expand to dwarf the space occupied by traditional GUIs [8].
4.2. Natural User Interfaces (NUI)

Natural User Interfaces are the interfaces that enable users to interact with a computer in the same ways we interact with the physical world, through using voice, hands and bodies. Instead of using a keyboard and a mouse (as is the case with GUIs), a natural user interface allows people to speak with machines, stroke their surfaces, gesture them in the air, dance on mats that detects the feet movements, smile at them to get a reaction, and so on. The naturalness refers to the way they exploit the everyday skills we have learned, such as talking, writing, gesturing, walking, and picking up objects [10].

Natural User Interfaces are interfaces which are natural to use and they are based on the nature or natural elements. It suggests the user is able to use the interface with little to no training. The elements of these interfaces are close to be invisible or become invisible with successive learned interactions. This is important as it reduces the cost of using the software; therefore suppress the need to train all the users to use it [8].

A NUI relies on a user being able to quickly transition from novice to expert. While the interface requires learning, that learning is eased through design which gives the user the feeling that they are instantly and continuously successful. Thus, “natural” refers to a goal in the user experience – that the interaction comes naturally, while interacting with the technology, and that the interface itself is natural.

The word natural is used because from the scenario using GUIs, computer interfaces use artificial control devices whose operation has to be learned; in the NUI perspective the word refers to the user’s behavior and feeling during the experience rather than the interface being the product of some organic process. This is very complex because it is a property that is actually external to the product itself.
The achievement of this state is the end result of rigorous design leveraging the potential of modern technologies to better mirror human capabilities. According to [8], this term natural should be faced as a design philosophy and source for metrics enabling an interactive process to create a product. The term should refer to the way users interact and feel about the product, or more precisely; what they do and how they feel while using it and not reduce its importance to something that essentially mimics the real world into

Therefore a successful NUI should be a product that mirror user’s capabilities, meet their needs, take full advantage of their capacities and fit to their task and context demands. And even due the level of engagement provides to the users enjoyment while using the interface.

According to [24], the interactivity in a NUI is based on simple skills, which means that the learned skills that only depend upon innate abilities. This limits the complexity of these skills, which also means simple skills are easy to learn and can be reused and adapted for many tasks without much effort. The learning process for simple skills is typically very quick and requires little or no practice to achieve an adequate level of competence. Many times learning can be achieved by simply observing someone else demonstrate the skill once or twice.

In the other hand, still according to [24], the GUIs relies on composite skills, which are learned skills that depend upon other composites or skills, which means they can enable users to perform complex, advanced tasks. It also means that relative to simple skills, composite skills take more effort to learn, have a higher cognitive load, and are specialized for a few tasks or a single task with limited reuse or adaptability. Composite skills typically require learning and applying specific concepts that tie together the requisite kills. In order to achieve an adequate level of competence with composite skills, both conscious effort and practice are typically required.

### 4.3. NUI for learning

NUIs can be considered an attractive paradigm of tools for education and training because of the potential to provide a sense of immersion in the learning environment to the student; since this paradigm is based on few elements on the interface and low gap for understanding and master the gestures and commands available, students can indeed focus on the content target of the class and improve the outcome learning. Even due the fact NUIs largely doesn’t support multiple open windows and files helps keep students focused on their work and not tempted to multitask with applications such as social networks.

Another advantage is the adaptive learning environment that responds as the learner interacts with it, allowing users to interact with virtual models and other objects. This set of features
tends to engage and stimulates students to follow actively the flow of information presented along a lecture.

Moreover, due its properties it could enhance the experience of educational tools for students who learn differently such as learners with autism or even for students that confronts physical challenges such as for learners with mobility challenges might use text commands to drive computer interactions.

The improvement of immersiveness when performing a task in such interface can be justified by means of cognitive load when interacting in a NUI. Cognitive load is defined on [24] as the measure of the working memory used while performing a task. The concept reflects the fact that our fixed working memory capacity limits how many things we can do at the same time. It states that certain types of learning exercises used up a significant portion of our working memory and since working memory is limited, the working memory is not available for the actual learning task. This means that learning is not as efficient when the exercises use too much memory. There are three different types of cognitive load intrinsic, extraneous and germane.

- **Intrinsic cognitive load**: the inherent difficulty of the specific subject matter to the student.

- **Extraneous cognitive load**: the load created by the design of the learning activities. For example, describing a visual concept verbally involves more cognitive load than demonstrating it visually.

- **Germane cognitive load**: it is actually used in the learning processing. The load is involved in processing and understanding the subject matter. Learning activities can be designed to help users understand new material by preferring germane load.

This theory claims that extraneous load should be minimized to leave plenty of working memory for germane load, which is used for learning. The inherent difficulty of a subject cannot be changed, but the current intrinsic load can be managed by splitting complex subjects into sub-areas.

As suggested on [24], this theory can be applied to human computer interaction field, and second them, composite skills depend upon several other skills and require conceptual thought, which generates extraneous load and reduces the working memory available for the intrinsic load of the actual task and the germane load for learning how to use the rest of the application. This results in users who are constantly switching between making high-level decisions about their tasks and figuring out the interface.
In contrast, simple skills have a low extraneous load because they are built upon the largely automatic innate abilities. This leaves most of the working memory available for the germane load of learning more advanced aspects of the interface as well as the intrinsic load generated by the actual task, which in the context of this work; the multimedia content. This means the users can focus on high-level thoughts and decisions and on getting better at what they do rather than on using the interface itself.

Diverse projects are being developed under NUI paradigms and offer to people innovative ways of interactivity beyond the traditional keyboard and mouse such as voice recognition, hands-free gestures and multi-touch interactions. Considering the purpose of this work, the touching devices (such as the ones for Tablets-PC) interactivity for multimedia learning are the most relevant to be investigated.

The interactivity on such devices can be divided into single and multi-touch gestures for interactions. Commonly single touch and drag actions can select and move (or draw depending on context). Zoom and rotation are handled with two fingered pinch/expand actions or twists. Further multi-touch gestures have yet to be standardized across programs.

According to [22], researchers have been exploring diverse aspects of cognition for educational purposes when performing physical manipulation on interfaces of digital devices. The studies on this area commonly involves aspects of embodied interaction, which essentially explores the relationship between more of our senses and in particular includes touch and physical movement that are believed to retain knowledge that is being acquired. Some educational approaches suggest that physical movement and touch enhance learning.

A study about including the haptic channel in a learning process [24] found that the immediate sensorimotor feedback that received through the hands can be transferred to working memory for further processing.

While learners process information in the visual and auditory channels in multimedia learning, by adding the haptic channel in the learning process provides those added benefits: they use hand movements to actively interact with the content and receive immediate sensorimotor feedback about causal interactions and functional relations among system entities, all of which are central to system understanding.

Still according to [24], interactions with information coming in through the haptic channel are different from our interactions with information entering the visual and auditory channels. Specifically, when incorporating the haptic channel in the learning process, learners not only can visualize the information but also interact with it through their hand controls. The immediate sensorimotor feedback they received from their hand motions can be transferred to working memory for further processing. A benefit of learning with kinematics displays is that it enables
learners to actively engage and participate in the meaning-making journey. It is a two-way communication process, with learners controlling the pace, speed, direction, and magnitude of the exploration.

4.4. Windows 8 as a NUI

NUI is being widespread along diverse applicatives, but they’re still far from being ubiquitous for a common user as it claims, and there’s one frontier that has yet to be conquered, the frontier to offer consistent solutions to users for their nowadays tasks [27].

Currently some software solutions running over hardware such as Tablets, Tablet-PCs and smartphones with multi-touch capabilities are the most diffuse and close to NUIs essence that are available to support people on their tasks every day.

Emerging from these solutions, it is noteworthy to refer to the most recent Windows version released by Microsoft: the Windows 8. This system introduces significant changes to the operating system's user interface, it provides a two interfaces paradigm, one based on the traditional desktop perspective and another focused on touchscreen interactivity, ideally for tablet devices.

Users that have been watching the evolution of Windows are undoubtedly well aware of the fact that the Windows 8 user interface has suffered a major overhaul, especially when the operating system is running on next generation form factors such as Tablet PCs or slates.

While Windows 7 was the first Windows client to enable users to add touch commands to classic input models such as the keyboard and mouse, it will most probably be the NUI innovations in Windows 8 to convince customers to embrace touch on traditional form factors such as desktops and notebooks, and not just on new devices [27].

According to [26] specifications, the applicatives running can be designed to user versatile input methods. Pointer APIs allow developers to work with any input modality (mouse, keyboard, pen, touch) without knowing the origin of the input data. Higher-level gesture APIs let developers to use pre-defined touch gestures and detect multi-touch events in your apps. There is also the possibility to build custom touch gestures or use input-specific APIs to provide a unique experience to the applicative.
Moreover, Windows 8 introduces a new programming paradigm: the Windows Store apps. These apps offer a clean, polished user experience that push app experiences to the forefront, and immerse the user in a full screen environment that's tailored to the user's hardware and context. These apps are optimized for touchscreen environments and are more specialized than current desktop applications. Apps can use "contracts"; a collection of hooks to provide common functionality that can integrate with other apps, such as search and sharing [28].
App contracts are a way for users to seamlessly search across and share content between unrelated apps. They extend the usefulness of the app by eliminating the need to work with varying standards or app-specific APIs to access data stored or created by another app, all while keeping users in your branded experience. Contracts are designed to offer essentially functionalities for searching information, sharing content, defining common settings and use external devices across the applicatives [26].

Through tiles on the Start screen, apps are alive with activity and can deliver vibrant content, even when they’re not running. Using live tiles, the app can provide useful data to the user, while minimizing battery usage. Windows Push Notification Services enables the app to receive messages and send them to the app’s live tile or provide a notification to the user.

The touch-optimized platform created allows that people interact with in an intuitive way. As a result, interactions with the operating system and UI elements feel natural and responsive. Animations and transitions are used deliberately, and effects are subtle and designed to enhance the connection between users and their PC. App layout and presentation are simplified with easy-to-use layout APIs and presentation controls.

![App with tiles features](image)

**Figure 14 - App with tiles features**

According to specifications on [26], the built-in animations library lets developers to create smooth, animated experiences from a comprehensive set of predefined animations that are
lively and unique. With 3-D transformations developers can add smooth, fluid, visual experiences like perspective transforms and flipping elements on and off the screen.

Flexible layout options make app layout easy and consistent with Windows 8. Semantic Zoom is a touch-optimized way to navigate through large content collections. Users can pan or scroll through their content, and then zoom in or out to view more or less information. This lets apps present content in a more tactile, visual, and informative way than traditional navigation and layout patterns like tabs.

Figure 15 - Semantic Zoom

According to Windows 8 specifications [26], the Sensors APIs provides resources for developers to build apps that are tailored to the user’s context, and apps that scale to support both compact and large form factors. The platform allows access input from eight sensors, including accelerometer, inclinometer, gyroscope, compass, ambient-light, orientation, and simple orientation of a device, and, with a user’s permission, geolocation

However, despite all this innovative features mentioned, the introduction of this new paradigm has become controversially among since common users as well as human computer interaction experts. According to a research done by [33] with PC experienced users testing Windows 8 on both regular computers and tablets found out some problems regarding this new paradigm.

The results of that research indicate that with the double desktop environment on a single device is a prescription for usability problems for several reasons:

- Users have to learn and remember where to go for which features;
• When running web browsers in both device areas, users will only see (and be reminded of) a subset of their open web pages at any given time;

• Switching between environments increases the interaction cost of using multiple features;

• The two environments work differently, making for an inconsistent user experience.

According to [33] one of the worst aspects of Windows 8 for experienced users is it no longer supports multiple windows on the screen and the main UI restricts users to a single window. The system does have an option to temporarily show a second area in a small part of the screen, but none of their test users were able to make this work.

When users can't view several windows simultaneously, they must keep information from one window in short-term memory while they activate another window. This is problematic for two reasons. First, human short-term memory is notoriously weak, and second, the very task of having to manipulate a window—instead of simply glancing at one that's already open—further taxes the user's cognitive resources.

Another aspect is that the Windows 8 UI is completely flat and there is no pseudo-3D or lighting model to cast subtle shadows that indicate what's clickable (because it looks raised above the rest) or where users can type (because it looks indented below the page surface).

There's a reason GUI designers used to make objects look more detailed and actionable than they do in the Windows 8 design. As an example given by [33] about it is demonstrated in the Figure 16 showing the settings menu:

![Figure 16 - Defining App setting over Charms](image-url)
It is not clear where users can click, everything looks flat, and in fact "Change PC settings" looks more like the label for the icon group than a clickable command. As a result, many users in the testing didn't click this command when they were trying to access one of the features it hides.

The results of [33] had shown problems with users overlooking or misinterpreting tabbed GUI components because of the low distinctiveness of the tab selection and the poor perceived affordance of the very concept of clickable tabs. Moreover users were either not relating to the icons or simply not understanding them.

Many other features are initially hidden and are revealed only when users perform specific and often convoluted gestures. For example, the users in test had great difficulty with an extraordinarily basic task: changing the city in the weather app. Obvious gestures, such as clicking the name of the current city to change locations, didn't work. Users' difficulties were exacerbated by the fact that the "Modern" GUI style doesn't indicate which words and fields are active and/or can be changed.

One of the most promising design ideas in Windows 8 is the enhanced use of generic commands in the form of the so-called "charms." The charms are a panel of icons that slide in from the screen's right side after a flicking gesture from its right edge (on a tablet) or after pointing the mouse to the screen's upper-right corner (on a computer).

In practice, according to [33] conclusions, the charms are not satisfactory for new users. The old saying, out of sight, out of mind, turned out to be accurate. Because the charms are hidden, our users often forgot to summon them, even when they needed them.

Furthermore, the charms don't actually work universally as Windows 8 claims because they're not true generic commands. In the [33] test, users often clicked Search only to be told, "This application cannot be searched." Enough disappointments and users will stop trying a feature since it violates basic usability guidelines; that is, you shouldn't tease users by offering a feature that isn't actually available. Finally, not all users understood that the commands are context dependent and do different things on different pages.

4.5. Conclusions

Resuming the most important conclusions found on discussions along previous chapters, regarding exclusively interface and interactivity aspects, it is possible to highlight the following requirements to be achieved for supporting the annotation process and multimedia learning in classroom environment:

- Interface offering low levels of cognitive effort to be learned and mastered;
Design and implementation of a Natural User Interface to support the annotation process for multimedia learning

Diogo Borges Krobath

- Students should focus on the multimedia content worked in class and suppress other interface elements;

- Interface should incentive students to manipulate content in order to avoid “entertain me mode”;

- Solution should support not only exclusively multi-touch interactivity but also include pen, keyboard and mouse as inputs.

Based on these conclusions and taking key properties brought by this chapter about NUIs such as low cognitive effort to learn and use the interface as well as immersion over the content being worked; it is natural to think as this interface paradigm as the most appropriate to be used in this work.

Moreover, considering the current spread and evolution of multi-touch devices and even the cognitive aspects of this mode of interaction for learning discussed in this chapter (by associating the haptic feedback when interacting with the content target to be learned by a student); it can be considered the ideal way of interaction to be used in our solution.

Another relevant aspect is the need of supporting multiple ways of interactivity and inputs (pen, mouse, keyboard and finger) in order handle the tasks; so considering the current devices the choice for a Tablet-PC as the ideal hardware to support the applicative target of this work.

Besides that, due the solid support of multiple ways of interaction and also all the resources presented on Windows 8 for increasing the user experience such as having fast and fluid transitions, intuitive interactions, content oriented in a full-screen and immersive interface design and also other factors related with the widespread of adoption of Windows and its API consistency to allow exchanging data with Polinotes components (described in details on implementation chapter) makes undoubtedly the adoption of Windows Store App as the most suitable environment to launch the applicative designed for this project.

However design choices have to be carefully considered while implementing the applicative. Due the problems found on Windows 8 adoption, the main issue of not being possible to handle multiple windows at the same time and even the negative user experience by switching applicatives through desktop perspective and Windows Store apps indicates that the software should provide all the resources and functionalities to keep the user within its environment while performing annotations.
The multimedia content presented in the lectures should also be incorporated inside the app functionality in order to avoid users to handle multiple windows and switch to Desktop paradigm to retrieve the slides objects; and besides that to facilitate the annotation process.

Considering the users issues while discovering the charms and even understanding the context concept introduced by Windows 8; it can be concluded that the core tasks done by students over the app shouldn’t strongly rely on these features and even for contexts where students are performing some activity that might be complemented with some resource from contracts or charms the app should suggest somehow their usage along the execution of the task.
5. Designing for the applicative

In this chapter are described the design decisions to support the implementation of Polinotes Student App. It starts by presenting the design principles for applicatives on Windows Store and proceeds by matching these principles with the requirements identified in previous chapters. It presents at the end the definition of set of gestures, commands and navigation patterns for the applicative.

5.1. Design for Windows Store Apps

The Windows Store is a digital distribution platform in Microsoft's Windows 8 and Windows RT operating system. This platform provides listings for desktop applications certified to run on Windows 8, but is also the primary distribution platform for a new type of app called Windows Store apps.

The following subsections describe the most relevant aspects of design, according to Microsoft [34], found on apps of Windows Store.

5.1.1. Navigation Patterns

Hierarchical system

It is considered the navigation most used by applicatives on Windows Store. This pattern is best for when there is large content collections or many distinct sections of content for a user to explore. It is based on hub navigation pattern, which means segregation of content into different sections and different levels of detail.
The pages of an applicative following this pattern can be divided into Hub pages, Section pages and Detailed pages. An example of these pages and their transitions are shown over the Figure 18.

- **Hub pages**: are the first pages to be shown to users, its content is displayed in rich horizontally panning view allowing users to get a glimpse of latest information updates. It consists of different categories of content, each of which maps to the app's Section pages. Each Section should bubble up content or functionality. The Hub should offer a lot of visual variety, engage users, and draw them in to different parts of the app. The Hub page pulls information from different areas of the application onto one screen.

- **Section pages** are the second level of an app. In these pages content can be displayed in any form that best represents the scenario and content the Section contains. The Section page consists of individual items, each of which has its own Detail page.

- **Detail pages** are the third level of an app. At this level details of individual items are displayed, the format of which may vary tremendously depending upon the particular type of content. The Detail page consists of item details or functionality. These pages may contain a lot of information or may contain a single object, such as a picture or video.
Flat system

This pattern is often seen in games, browser, or document creation apps, where the moves between pages, tabs, or models that all reside at the same hierarchical level. This pattern is recommended when the core scenario involves fast switching between small number of pages or tabs. The Flat system separates the content into different pages.

This approach usually uses a top bar for switching between multiple contexts. This bar is a transient element that resides at the top of the screen, and is made visible when users swipe...
from the top or bottom edge. The format how items are in the bar can vary; a typical treatment is the use of a simple thumbnail.

Unlike the hierarchical system, there is typically no persistent back button or navigation stack in the flat system, so moving between pages is usually done through direct links within the content or the top app bar.

![Figure 20 - Navigation using the flat pattern](image)

5.1.2. Commanding patterns

Canvas

Canvas is the area where the main actions takes place; it is aimed to allow users to complete the core scenarios of the apps through directly manipulation with content rather than using alternative commands that act on the content. On the canvas further controls such as for filtering, pivoting, or sorting takes their actions.
Navigating with the edge swipe

Users can navigate within apps and throughout the system by swiping a finger or thumb from an edge. In order to use Windows Store apps appropriately the following swipes were defined as the standards:

- Swiping from the bottom or top edge of the screen reveals the navigation and command app bars.

- Swiping from the right edge of the screen reveals the charms that expose system commands.

- Swiping from the left edge cycles through currently running apps.

- Sliding from the top edge toward the bottom edge of the screen closes the current app.

- Sliding from the top edge down and to the left or right edge snaps the current app to that side of the screen.
Charms

The charm contains a set of commands potentially executed or defined in common among user’s applicatives. This set is composed by:

- **Search**: which is intended to allow users to search through applicative content from anywhere in the system, including other apps.

- **Share**: with purpose to let users to share content from his applicative with other people or applicative. Its purpose is also designed for functionalities to receive shared content.

- **Devices**: it is associated with stream of audio, video, or images from the applicatives to other devices in their home network.

- **Settings**: Consolidate all of settings under one roof and let users configure the applicatives with common mechanism they’re already familiar with.

![Command to enable the Charms](image)

Figure 23 - Command to enable the Charms

The app bar

The app bar is placed on the bottom and is designed to be used primarily as a commanding surface, but it can also be used to alter the way in which content is being viewed. Switching views, pivoting, filtering and sorting can all be done by using the app bar. All app bar items should act on the content currently in view. The commands shown are relevant to the user’s context, usually the current page, or the current selection.

The bar is not visible by default and appears when a user swipes a finger from the top or bottom edge of the screen. It can also appear programmatically on object selection or on right click. This resource goes away after the user taps a command, taps the app canvas, or repeats the swipe gesture.
The top bar

The top app bar is used primarily for navigating sections or pages of an app that use the Flat navigation pattern. Sometimes called a navigation app bar, it can also be used along with the Hierarchical pattern, instead of the header menu, to provide global navigation controls. The top app bar should show up on every page and at all levels of the app to provide users with a convenient, consistent way to get around.

Context menus

The context menus can be used for clipboard actions (like cut, copy, and paste), or for commands that apply to content that cannot be selected (like an image on a web page).

The system provides apps with default context menus for text and hyperlinks. For text, the default context menu shows the clipboard commands. For hyperlinks, the default menu shows commands to copy and to open the link.

5.1.3. Touch interaction patterns

The Windows 8 has a set of touch interactions that is standard along the system. By using it properly it helps the user to learn how to use the app since it is intended to appear similar to interact as it is with the outside environment. The following list describes the standard touch interactions defined by Microsoft.
Press and hold to learn

This touch interaction causes detailed information or teaching visuals (for example, a tooltip or context menu) to be displayed without a commitment to an action. Anything displayed this way should not prevent users from panning if they begin sliding their finger.

![Figure 26 - Press and hold to learn](image)

Tap for primary action

Tapping on an element invokes its primary action, for instance launching an application or executing a command.

![Figure 27 - Tap for primary action](image)

Slide to pan

Slide is used primarily for panning interactions but can also be used for moving, drawing, or writing. Slide can also be used to target small, densely packed elements by scrubbing (sliding the finger over related objects such as radio buttons).

![Figure 28 - Slide to pan](image)
**Swipe to select, command and move**

Sliding the finger a short distance, perpendicular to the panning direction, selects objects in a list or grid. Display the application bar with relevant commands when objects are selected.

![Swipe to select, command and move](image)

*Figure 29 - Swipe to select, command and move*

**Pinch and stretch to zoom**

While the pinch and stretch gestures are commonly used for resizing, they also enable jumping to the beginning, end, or anywhere within the content with Semantic Zoom. A Semantic Zoom control provides a zoomed out view for showing groups of items and quick ways to dive back into them.

![Pinch to stretch and zoom](image)

*Figure 30 - Pinch to stretch and zoom*

**Turn to rotate**

Rotating with two or more fingers causes an object to rotate. Rotate the device itself to rotate the entire screen.

![Turn to rotate](image)

*Figure 31 - Turn to rotate*
**Swipe from edge for app commands**

As described in commanding section, app commands are revealed by swiping from the bottom or top edge of the screen, swiping from edge for system commands and swiping from the left edge cycles through currently running apps.

**5.1.4. Touch posture**

Designing for touch goes beyond of designing the content presented on the screen. It requires designing for how the device will be held (grip). Typically, different people have a few favorite grips when holding a tablet.  

The current user task and how it should be presented usually determines which grip is used. However, the immediate environment and physical comfort also affect how long a grip is used and how often it’s changed.

Considering interactions aspect, due slates are most often held along the side, as shown in the Figure 32, the bottom corners and sides are ideal locations for interactive elements.

![Figure 32 - Interaction areas](image)

Regarding the reading areas, as shown in the Figure 33, content in the top half of the screen is easier to see than content in the bottom half, which is often blocked by the hands or ignored.
According to Microsoft [34], the most common grips used to hold a tablet are:

**One hand holding, one hand interacting with light to medium interaction**

This grips offers right and bottom edges with fast interaction and due limited reaching makes touching more accurate, however, as show in the Figure 34, the lower right corner might be occluded by hand and wrist. It is ideal for reading, browsing and light typing.

**Two hands holding, thumbs interacting with light to medium interaction**

This positioning produces quick interactions for Lower left and right corners and increased accuracy for touching due thumbs are anchored. Since everything in the middle of the screen is difficult to be reached, touching middle of screen requires also changing of posture. This grips is recommended for reading, browsing and tasks demanding light typing.
Device rests on table or legs, two hands interacting with light to heavy interaction

It allows fast interaction for bottom of the screen, since the need for reaching is reduced it makes touching more accurate. The main disadvantage is occlusion of lower corners by hands and wrists. Unlike previous cases, this is considered ideal for heavy typing due hands are closer to the bottom.

Device rests on table or stand, with or without interaction

In this case only bottom of screen offers at some way satisfactory interactions because considering that touching top occludes content and might also knock a docked device off balance. The distance reduces readability and accuracy which demands increase of targets for touching for designs with this grips purpose. This is ideal for watching movies and listening music.
Besides this analysis of common grips it is worth to mention that most people are right handed, which implies that most people hold a slate with their left hand and touch it with their right. In general, elements placed on the right side are easier to touch, and putting them on the right prevents occlusion of the main area of the screen.

![Right hand interaction](image)

**Figure 38 - Right hand interaction**

### 5.1.5. Touch targets

This is a challenge topic for multi-touch design, since dimensions of fingers varies considerably (the average adult finger is about 11 millimeters (mm) wide, while a baby's is 8 mm, and some basketball players have fingers wider than 19 mm) and the screen size to shown it is in general smaller than traditional laptops [34].

According to Microsoft, a good minimum size recommend for designing targets is 7x7 mm for the cases when a mistake can be corrected in one or two gestures or within a tolerance of 5 seconds. It is also recommended to have a padding of at least 2 mm among touch targets.

![Minimum size recommended for touch targets](image)

**Figure 39 - Minimum size recommended for touch targets**

For the cases when touching is devoted for a more critical operation such as close or delete, it is recommended to use 9x9 mm targets instead. This case turns to be suitable when touching a wrong target takes more than two gestures or more than 5 seconds to correct the mistake.
For extreme situations, when it is really need to reduce sizes in order to make a target fit into a layout, it is considered of to use 5x5 mm targets as long as touching the wrong target can be corrected with one gesture. Using 2 mm of padding between targets is extremely important in this case.

As discussed, different sizes work for different situations. Actions with severe consequences (such as delete and close) or frequently used actions should present large touch targets in order to minimize the effects of missing taps. Actions that are not much performed which also offers minor consequences can be designed as small targets. The relationship between target size and number relative of missed taps can be seen in the Figure 42.
5.2. Basic actions for the design of annotation process

This section will proceed with the analysis made on previous chapters about the annotation process and multimedia learning. As presented in the chapter 3, the most relevant elements found in notes along diverse softwares supporting annotation process are:

- Handwriting
- Digital text
- Illustrations
- Page

The following subsections discuss the most relevant conclusions regarding those elements and how their basic functionalities should be incorporated into Polinotes Student App design in order to improve the way students perform annotations.

5.2.1. Handwriting

Resuming the main issues when writing on paper as problems of space while performing annotations and, due high effort on duplicating content already place on a paper, the possibility to fatigue students along classes; it turns out to be essential actions such as moving content, resizing and duplication. Also It can be considered useful allow rotations and simple operations for formatting (such as changing color or any other pattern in the composition of the line) in order to perform few adjustments

Besides that, the action of writing should be as much as possible similar to the user experience obtained when writing on a paper in order to suppress the step of learning and getting used to write with an electronic pen.

It can be observed that the software should also offer basic mechanisms to differentiate among different words written in order to make less time consuming the selection words to perform some action in sequence. Functionalities like OCR recognition are welcomed for such software purpose.
5.2.2. Text

Considering a solution where students use a multi-touch and pen methodology, the main way of typing texts would be through a virtual keyboard. This representation of real keyboard in general is recommended for light typing since it doesn’t allow a convenient positioning for hands as a physical keyboard do. There are still some techniques to suppress this problem such as suggestion of words or even enhancing the touch target on some keys more often typed based on previous content partially typed.

However considering the scope of annotations, it can be expected that students should use less this element in order to add new content to his notes, which is also strongly dependent on how well the handwritten input is in the solution.

In the other hand, due scenarios where the content presented in class could be brought automatically to students notes or even the possibility to use OCR recognition to transform handwritten input into digital text, the action of edit a text is expected to be done more often.

As perceived in the previous chapter the edition of texts offers some challenges for design, first to perform an accurate selection of the fragment of text to be edited and second due changes of states in the interface that might cause a bad user experience.

Regarding problems of selection some techniques like increasing the size of text when in edition mode or using the pen as instrument to determine start and final positions of texts instead of using fingers can suppress these problems.

Since texts demand basic operations such as rotation, move, resize, duplicate, application of basic formatting (bold, underline, italic, change of color etc) the way change of states is designed contributes directly the efficiency of interaction with this element.

Despite the multi-touch and pen interactivity considerations, as discussed in previous chapters, a software solution should also be able to provide input text through a traditional keyboard and mouse. The modality using a physical keyboard demands a screen positioning close to the way people use in traditional Desktop personal computers while in virtual keyboard by means of multi-touch interactivity demands a positioning similar to the ones described in the previous sections of this chapter. Therefore a choice for a device such as Tablet-PCs turns to be the most appropriate to handle these different modalities of inserting digital text into the annotations.
5.2.3. Illustration

Illustration in this context is related with images obtained over lecture content, graphics imported by student from other resource or basic draws performed in order to complement some content.

Since annotation is a time constraint task and in general drawing this element is time consuming and highly dependent of student’s abilities, most part of this element should be provided previously or at same time as lecture goes on.

In respect to that, and also regarding the previous analysis where problems with space constraints were determinant to structure the annotation and consequently contribute to its quality, it can be deducted that the most frequent operation to be performed on this element is moving and place in different places along the notes document in order to adjust a better configuration according the way student structured it.

Other basic operations over this element are rotation and resize, which aims a better adjustment with the context, and duplicate case the student wants to use the same illustration to support the annotation over the same or other pages of the document.

5.2.4. Page

Since the result of the annotation of a student is a collection of pages, the design for such software has to include an efficient way to navigate along elements of this collection and manipulate a single element of the collection.

Besides navigation and manipulation, it has to offer mechanisms to edit the other objects already placed over the page. This situation can be considered a design problem and based on software solutions experimented by this work, this situation is usually solved by imposing different states for the interface such as edition mode, where user can edit content of elements on the page and manipulate the page itself (scrolling up/down and performing zoom in/out), and moving mode where the elements are not editable but in a way to facilitate the operation of moving.

An interesting alternative for solving this problem would be design this element in two states reproducing the main functions of annotation described on Chapter 2: encoding and storage functions. On encoding function, differently from the edition state just discussed, the users can’t manipulate the page but they are able to edit and move all the content placed on the page and it would be the default mode, not relying in any previous switch state, in this case the page would be truly and metaphor of a physical page where there are even the physical limits of
placing content and the user doesn’t need to perform manipulations in order to work on it. The second state would be the associated with storage function, therefore focusing on features to visualize the page and browse among different pages.

This element should support also rotations, as described in previous section, user can use a tablet in different positions and since each position offers its advantages and disadvantages and therefore the student might use different orientations for performing diverse tasks related with his notes.

The table 3 summarizes the actions identified as essential to be placed for each element of the annotation process.

<table>
<thead>
<tr>
<th>Actions\Elements</th>
<th>Handwriting</th>
<th>Text</th>
<th>Image</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Rotate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Resize</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Duplicate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Basic formatting</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change content</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Navigate over collection</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Delete</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 3- Actions and elements of annotation

5.3. Polinotes Student App design

The following subsections describe how the actions mapped in previous section are designed in the applicative in terms of gestures and commands.

5.3.1. Gestures design

According to [8], a gesture in touch screen devices can be thought of as a function call, the user selects the function at registration phase, and during continuation phase selects specifies the parameters of the function; the function is executed at termination phase.
Summarizing a gesture consists in three stages:

- Registration: moment when the type of action is set;
- Continuation: adjusts the parameters of gesture;
- Termination: when gesture ends;

Another state “Out of range” can be included and it means when no touch interaction is introduced.

![Figure 43 - Stages of a gesture](image)

**Rotate**

This gesture can be applied to perform the action of rotation for handwritten, texts and illustration. Basically it is designed to be performed by placing two fingers over the object and rotate.

However, for the purpose of rotation of pages a better solution could be addressed by analyzing device orientation and then consequently automatically rotate according to that orientation instead of depending on users gestures to perform this action.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate an object</td>
<td>Place two fingers on an item</td>
<td>Turn the fingers</td>
<td>Lift any finger</td>
</tr>
</tbody>
</table>

*Table 4- Rotate gesture*
Resize

This gesture is applied to handwritten, texts and illustrations. It should be performed by applying proportional changes, both in width and height on those elements, according to the way user is moving his fingers, if it is increasing their distance the movement should enlarge the object and similarly if they are reducing the distance it should reduce the dimensions of the object.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enlarge or reduce an object</td>
<td>Place two fingers on an item</td>
<td>Move by reducing or increasing the distance of the fingers, pinch or enlarge then resizing according with movement</td>
<td>Lift any finger</td>
</tr>
</tbody>
</table>

Table 5 - Resize gesture

Move

This gesture is responsible for covering the moving actions for handwritten, texts and illustrations. As discussed before, this is one of the most important operations done by a student along annotation process and due this importance it should be performed simply as placing a finger on the object and moving to the desirable place.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move an object</td>
<td>Place a finger on an item</td>
<td>Move the finger around on the surface</td>
<td>Lifts the finger</td>
</tr>
</tbody>
</table>

Table 6 - Move gesture

Change content

This gesture is applied only for texts, it is done when user double tap the text element and it enters on an editable state. In this state other gestures or mechanisms can be incorporate in order to allow users to properly change. To go away from the editable state, the user has to double tap anywhere outside the object.

For the context of pages, by default a page is able to have content changed, therefore users are able to add and manipulate content, however in order to avoid conflicts with other gestures or even false positives for changing mode that could confuse user, the other mechanisms will be provided to enact content changing rather than being performed by gestures.
Design and implementation of a Natural User Interface to support the annotation process for multimedia learning

Diogo Borges Krobath

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the content of an object</td>
<td>User places a finger down</td>
<td>Performed in short time</td>
<td>User lifts the finger</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1- User lifts his finger</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2- User places again a finger down in the same position before</td>
<td></td>
</tr>
</tbody>
</table>

Table 7 - Change content gesture

Select part of content (text)

This gesture is specific for texts and it aims to select fragments in order to support other operations to change content or format a specific part of text.

The action of determining fragment of texts shouldn't be restrict to this gesture, as discussed before, selecting specific positions based on touch interactions might not be accurate, therefore this operation should also be supported by virtual keyboard or even pen pointer.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select part of a text in order to apply an operation</td>
<td>Place a finger on a starting point</td>
<td>Move the finger in direction to the ending point</td>
<td>User taps somewhere outside the object</td>
</tr>
</tbody>
</table>

Table 8 - Select part of content

Next / Previous document

These gestures are designed to perform the action of navigation along pages. Due the chance of this gesture be highly performed as a false positive because the nature of others gestures defined for this project, it will be only available to users when he is visualizing and browsing along his notes and when students are in process of producing notes, other elements in the design should provide the function of transition between pages.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to the next page</td>
<td>Place a finger on a page</td>
<td>Move the finger to the left direction</td>
<td>User lifts the finger</td>
</tr>
<tr>
<td>Move to the previous page</td>
<td>Place a finger on a page</td>
<td>Move the finger to the right direction</td>
<td>User lifts the finger</td>
</tr>
</tbody>
</table>

Table 9 - Next/previous gestures
Since it is highly important offer to users efficient ways of moving elements around in order to be able to create spaces to quickly add content or structure his notes properly while a lecture is going on, it turns out to be essential define more gestures to manage collection of elements and move content around.

Therefore, besides all the basic gestures defined before, it is going to be presented few more gestures with the purpose just described.

Selection inside a rectangle

This method for selection is used when there are a considerable number of objects in the same area, so the user can essentially draw a rectangle that involves the target area. At the end all objects belonging to that area should be in selected state.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select objects by making a rectangle</td>
<td>User places a finger on an empty space</td>
<td>User moves the finger to the direction of farthest object to be included</td>
<td>User lifts a finger</td>
</tr>
<tr>
<td>around the target element</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10 - Selection inside rectangle gesture

Tap to select

This method of selections is designed for objects that are distant from each other or few objects in the same region. It consists in performing the tap gesture on each target to object to select them.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select the objects of interest by</td>
<td>User places a finger down</td>
<td>Finger doesn’t change position and remains down for not much time</td>
<td>User quickly lifts his finger</td>
</tr>
<tr>
<td>performing tap gesture on each object</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11 - Tap to select gesture

For these selection gestures, a visual feedback should be provided to users in order to make them understand which objects are selected and have the correct perception about the consequences of his actions over his notes.
The selection mode of objects should be over when user performs double tap on an empty part of the page.

Recapping the needs for creating spaces for keep the processes of annotation going on well, the following gestures are designed to achieve this issue.

**Change distance among objects**

Basically in this gesture the user touches the objects he wants to create space in between and then perform the movement of reducing the distance between the fingers. The correspondent is also possible to be done in case user wants to increase the distance among objects by also increasing the distance of his fingers placed over the objects on the touchscreen.

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase distance</td>
<td>1 - User places a finger on one object</td>
<td>Move by increasing the distance of the fingers</td>
<td>User lifts any finger</td>
</tr>
<tr>
<td>among objects</td>
<td>2 – User places another finger on another object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduce distance</td>
<td>1 - User places a finger on one object</td>
<td>Move by reducing the distance of the fingers</td>
<td>User lifts any finger</td>
</tr>
<tr>
<td>among objects</td>
<td>2 – User places another finger on another object</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 12 - Change distance among objects*

**Create / Reduce space**

This gesture is similar to the previous but rather than placing fingers on objects, they are placed on an empty space on the page.

Since from user perspective this movement of creating space is strongly related with paper behavior, it should carry also some specific situations such as when due this gesture objects get closer to the bottom boundaries of the page; for this situation an action of moving the elements to the next page will be taken. This approach will be brought back in details along the implementation description.

Similarly the action of reducing space can be performed in the correspondent manner by reducing the distance between fingers on the touchscreen.
Table 13 - Create/Reduce space gesture

<table>
<thead>
<tr>
<th>Logic action</th>
<th>Registration</th>
<th>Continuation</th>
<th>Termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create space on a region of the paper</td>
<td>User places two fingers down on an empty space of the page</td>
<td>User enlarges vertically the distance between those fingers</td>
<td>User lifts any finger</td>
</tr>
<tr>
<td>Reduce space on a region of the paper</td>
<td>User places two fingers down on an empty space of the page</td>
<td>User reduces vertically the distance between those fingers</td>
<td>User lifts a finger</td>
</tr>
</tbody>
</table>

5.3.2. Interface modes

From the gestures definition, next stage is to establish in which contexts each one of these gestures will be available to the user.

Defining the states of the interface had considered two main goals; first allow users to perform essential actions needed to carry the annotation process on the same state, therefore not wasting time changing states and aiming to provide an immersive and natural experience into the task user is performing.

The second goal is to allocate gestures together where they have clear purpose in terms of functionality and also evaluating their distinction regarding initiation and continuation states in order to avoid confusions and reduce the occurrence of false positives.

In order to attend these goals, the object page will be separated into two state modes:

- **Annotation mode**: aims to support the encoding function of the annotation; therefore it is where all the actions regarding the annotation process should be performed;

- **Visualization mode**: supports mainly the store function of the annotation; it contains all the functionalities to allow students to browse and manipulate their notes;

Analyzing the objects on annotation mode it is can be defined two more modes: manipulation and edition modes. Excepted from the object text, all others can be designed under manipulation mode since all the gestures that cover the actions of these objects are executed in the same way. Only object text should be designed for having an editable mode.
• **Manipulation mode**: this is the object mode that provides interaction with elements over the page. It is supposed to cover basic action of moving, rotating and resizing elements;

• **Editable mode**: it comprises actions of editing content placed in the text element

The table 14 represents the allocation of the gestures along these interface modes.

<table>
<thead>
<tr>
<th>Annotation mode</th>
<th>Manipulation mode</th>
<th>Editable mode</th>
<th>Visualization mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Resize</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Move</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Change content</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select part of content (text)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Next / Previous document</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Selection inside a rectangle</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tap to select</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change distance among objects</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Create / Reduce space</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 14 - Interface modes

5.3.3. **Commanding design**

It is clear that the set of gestures just defined doesn’t cover all the basic needs for the annotation process, the actions that still need some solution are:

• **Delete**: it is the action of removing an object from the interface;

• **Duplicate**: it performs copy of an object and consequently options to where this copy should be placed in;
- **Basic formatting**: functions to edit foreground color and more specifically for text objects options to change formatting such as bold, italic, underline etc.;

These actions have in common a property of being able to be applied over a single object or a group of objects previously selected (not necessarily all elements of the same object) and also should be performed only during annotation mode.

Therefore for these actions so far uncovered the applicative should provide tools to perform those commands and considering the Windows Store environment, a convenient approach is addressing those commands to the app commanding bar.

This implementation of these commands over Windows Store app commanding bar should present icons for issuing those commands respecting the context of the applicative; it only makes sense to perform any of those actions when this applicative is in annotation mode and after a previous selection on canvas.

### 5.3.4. Navigation design

The navigation needs for this project demands a scenario that could represent an affordance of the real world where content of notes is separated into different pages and the user can perform easily fast switching between those pages.

Back to the patterns of navigation natively supported on Windows Store apps, the most appropriate for handling this project turns out to be the flat system.

This approach uses the top bar for switching between multiple pages though. This bar should offer commands to conveniently navigate between pages that are close among them or even large set of pages, so for the last case be able to browse and preview the content while browsing, as it happens handling a physical sheet of paper.

For the case when user is in visualize mode, the gesture of moving next or previous document is also appropriate for this pattern of navigation.
6. Polinotes Student App implementation

This chapter describes the process of implementation carried out from the design described in the previous chapter. As starting point, it presents the software Polinotes as part of the solution and continues describing the development and design implementation of the applicative on Windows Store platform acting as a client of Polinotes package.

6.1. Polinotes

Polinotes is also part of CATS projects, it was developed by ARCSLab (Adaptable, Relational and Cognitive Software environment) of Politecnico di Milano and it aims to support the process of annotation and learning through multimedia content. It provides functionalities to allow students to take notes along the lectures, retrieve the multimedia content shown over professor’s presentation and send to students by allowing them to modify and interact with content instances, integrating at the same time with their annotations.

6.1.1. The choice for Polinotes

As discussed in the previous chapters, during the annotation process students tend to prefer using printed version of the multimedia presentation and also simple white sheets combined with printed content rather than a computer based solutions. The main reason for the group that choose to write over content of printed slides is due the possibility to record information having a precise reference with content provided by the teacher and to reduce the workload of reproducing content to the notes. On the other hand, the students that uses a simple white sheet claims that due free space it is more appropriate for performing notes with a customized structure and it contributes to the process of development of personal ideas.
According to the author of Polinotes [1], this software provides a solution for these needs of annotation: extracting multimedia content from professor’s slides presentation, send them automatically to students and enable them to organize and structure their notes using also the content just brought. The Polinotes software connects direct and automatic multimedia presentation shown by the teacher and the notes of the individual student.

Considering the teacher perspective, [1] claims that the professor can move freely between several multimedia presentations and viewing the slides in random order. The process of sending content is automatic and does not limit in any way the freedom of the speaker.

On Figure 44 it is shown the concept of Polinotes; by using the wireless coverage of the university is possible to establish a direct link between the student and the teacher in a way that the content displayed by the teacher during the multimedia presentation is replicated automatically on Tablet PC of a student.

Based on these features it is possible to conclude that Polinotes is an appropriate solution to support the functionalities of extracting and receiving multimedia content from professor’s presentation. Therefore the applicative target of this work should consist on a client able to connect to Polinotes package, receive the content and also enable students to interact with this content while performing their annotations.
6.1.2. Components of Polinotes Package

Polinotes consists on different components distributed in a classroom environment. The package is composed by three different softwares: Polinotes Teacher, Polinotes Students and Polinotes Discovery.

Polinotes Teacher

This component is essentially a plugin embedded on Microsoft Office Power Point 2010 and it is designed for professors that present the slides in public in any type of computer connect within a classroom network. This software acts as a server and has the purpose of providing the multimedia content to the client applications connected.

Polinotes Student

It is the client software and it is designed to take advantage of the content broadcasted from Polinotes Teacher. This component works as a plugin running on Microsoft Office One Note 2010 and due that it gives the possibility to students to use all the resources under Microsoft Office environment; this fact associated with the content automatically received from professor’s slide and integration with their handwritten notes with the stylus of the Tablet-PC claims to offer a powerful applicative to support the annotation process.

Polinotes Discovery

This component is basically a web service with the purpose to support the process of discovery of the IP Address where Polinotes Teacher is placed. Therefore using this component clients are able to connect easier and receive the content being presented.

6.1.3. Connectivity pattern of Polinotes

All components of the package Polinotes exploit the connection to a wireless network to interact and exchange the information.

In terms of network architectural, the fact that the process server Polinotes Teacher sends multimedia content about what is being displayed on the slide show to instances of Polinotes
Student willing to receive this content suggests an architectural pattern similar to a publisher-subscribe model.

The publisher-subscribe model is a paradigm of interaction that involves two main entities: the producers, that eventually publishes something and consumers that subscribe to receive some information from the producer. This paradigm allows consumers to express their interests to a certain type of event with the aim to receive a notification each time that the event to which it has been expressed interest happens.

In the publish-subscribe pure architectural pattern, shown in the Figure 45, the entity publisher is not aware of the identity of subscribers because the event notification to the individual subscriber is carried out by a third party called event notification service, which guarantees the decoupling between publishers and subscribers.

For the context of Polinotes, the publisher is the instance of the software Polinotes Teacher, which runs on the teacher's computer, the subscriber instances of the software Polinotes Student, running on Tablet-PC students, the event is unique and is to display the next slide during the multimedia presentation.

The communication between publisher and subscriber is direct on Polinotes and the component Polinotes Teacher sends to each instance of Polinotes Student data in the multimedia slide as soon as it appears during the multimedia presentation. Polinotes Teacher sends to subscribers a notification with the textual content and the presence of multimedia data to be downloaded by them.
6.1.4. Complete usage scenario

This section describes the operation and communication that takes place between the different components of Polinotes during a lecture. In this scenario, the teacher shows his multimedia presentation software using Microsoft Power Point 2010 with the support of the application Polinotes Teacher. Students take notes of the content using a Tablet-PC and the application Polinotes Student Notes in association with the software Microsoft One Note 2010. Tablet PCs are equipped with active digitizer, capable of recognizing the traits of digital ink drawn. Despite the role played by the service Polinotes Discovery in the early stages of connection.

For simplicity it is analyzed the interaction between an instance of the software Polinotes Teacher and only one instance of the software Polinotes Student, as if it was only one student who attends the class and uses the package Polinotes (see Figure 47). The communication procedure remains unchanged even for a higher number of client applications within the system.

In order to start a lesson, professor starts on his computer an instance of Microsoft Power Point 2010 with Polinotes Teacher appropriately installed. Accessing the Polinotes software on Power Point the professor can define the moment to start the lesson; at this moment, professor inputs his name and password and request to start a lesson. The software automatically connects with Polinotes Discovery, located on a server within network in order to register the IP address of professor’s computer. Polinotes Discovery returns to Polinotes Teacher a confirmation indicating the process occurred successfully.

In other hand, the student using a Tablet PC, open Microsoft One Note 2010 with Polinotes Student installed. Using this software, the student can inform the professor he wants to receive a multimedia presentation.
Based on the name of professor and a password informed, Polinotes Student requests to Polinotes Discovery the IP address correspondent with the instance of Polinotes Teacher. The service responds to the request by sending to Polinotes Student the IP address it can connect for receiving the multimedia content of the lesson. In terms of implementation, the process of connection among the component of Polinotes is described in UML diagram at Figure 48.

Once Polinotes Student has the IP address of professor’s computer, it establishes a direct link to the instance Polinotes Teacher that is running the multimedia presentation. After establishing the connection, Polinotes Teacher deals with the task to extract the content of the multimedia slide shown during the lecture and send them all to instances of Polinotes Student connected at that moment.

The UML sequence diagram at Figure 49 represents the activities performed between Polinotes Teacher and an instance of Polinotes Student after they are connected.

When the start of a slide show on Microsoft Power Point is performed, it raises an event that is handled inside Polinotes Teacher and after concluding some preliminary steps internally it initiates the extraction of content in the current slide by using sendData().
According to content in the multimedia slide the mode of transmission is different. When the content is text-based, Polinotes Teacher invokes an asynchronous callback method SendText() to send the text and its formatting information. Polinotes Students receives it and insert the block of text inside the application Microsoft One Note.

For the case the content is a movie, audio or image the method addMedia() is invoked in order to notify the software Polinotes Student about a need to proceed with the download of the multimedia content. The download is initiated by Polinotes Students when it invokes back the method download() exposed to Polinotes Teacher.

The transfer of media doesn’t affect the process of annotation made by the student and only when the download is concluded it is integrated to the document of notes on Microsoft One Note.

After exposure of first slide multimedia, the professor can continue and show the next slide of the presentation. This transition raises the event slideShowNextSlide within Polinotes Teacher. The software extracts all the content of currently displayed slide and in the same way as
described before sends the notifications to Polinotes Students that handles it manages the integration of content received within the digital notes.

The communication between Polinotes Teacher and Student doesn’t produce delays for both professor and students connected. The professor can normally use Power Point during the execution of PoliNotes and in the same way the student can write down his notes with Microsoft One Note.
6.1.5. Technology for connectivity

Since it is already known the way the package Polinotes works, it is relevant for this project understand the technology behind the Windows Communication Foundation (WCF) API, which is the responsible for the connectivity among the instances of components of Polinotes.

6.1.5.1. Windows Communication Foundation

WCF is the Microsoft unified programming model for building service-oriented applications. A WCF Service is a program that exposes a collection of Endpoints. Each Endpoint is a portal for communicating with the world. A Client is a program that exchanges messages with one or more Endpoints. A Client may also expose an Endpoint to receive Messages from a Service in a duplex message exchange pattern.

Each endpoint consists of address, binding and contract:

- **Endpoint Address** is a network address where the Endpoint resides; therefore the address to which the service responds;

- **Endpoint Binding** specifies how the Endpoint communicates with the world including things like transport protocol (e.g., TCP, HTTP), encoding (e.g., text, binary), and security requirements (e.g., SSL, SOAP message security);

- **Endpoint Contract** is placed at a higher level and is the software interface that make the service public, it is a collection of operations that specifies what the Endpoint communicates to the outside world. Each operation is a simple message exchange, for example one-way or request/reply message exchange.

The contract of a service can be of different types:

- **Service Contract**: define the API that the service provides. This type of contract is defined as an interface. NET that must be implemented by the developer with the logic of the service.

- **Data Contract**: define the types of data that are public and visible to third-party applications that use the service.

- **Message Contract**: define what data should be included in the header and what should be included in the body of the messages that are exchanged with the service.
This framework allows building service-oriented applications easily and maintaining the characteristics of efficiency and scalability required for the distributed applications. A key feature of the framework is the interoperability with other service-oriented technologies. It Compatibility with other technologies encourages the reuse of software pre-constructed and allows WCF applications to communicate with older software [16].

Therefore in order to enable the applicative developed for this project to connect and exchange data with Polinotes it is necessary to understand how this framework is implemented along its components and later reproduce a similar behavior of Polinotes Student on WCF framework for connecting to Polinotes Teacher, receiving and requesting multimedia content.

The following subsections cover, based on the description of Polinotes developer [1] and also analysis done on the source of the project, how WCF is implemented on each Polinotes component.

### 6.1.5.2. WCF implementation on Polinotes Teacher

The software architecture on Polinotes Teacher has four different packages within of the implementation (see Figure 50)

- Contracts;
- Service;
- Power Point Manager;
- Data.

The package Contracts contains all the interfaces defined as ServiceContract with which the software Polinotes Teacher communicates. The interfaces IDataInterface and ITeacherInterface are implemented within Polinotes Teacher because they are exposed to the outside during application execution; the others interfaces are used to invoke operations on Polinotes Student and Polinotes Discovery.

The interface ITeacherInterface handles operations of subscription and authentication of the client to the application software Polinotes Teacher. Respective roles "Subscribe" and "Unsubscribe" are invoked by instances of Polinotes Student to register the list of connected users inside of Polinotes Teacher and for removal from the list of clients. The method "Authentication" allows clients to authenticate using a password, which is decided at the beginning of the session by the teacher.
IDataInterface allows clients connected to downloading multimedia content integrated into the current presentation, so that you can integrate into digital notes of each student.

IStudentsInterfaceCallback allows the software Polinotes Teacher to invoke the execution of functions on the instances of Polinotes Student to ensure two-way communication.

---

**Figure 50 – Software architecture of Polinotes Teacher**

DiscoveryServiceI is used instead to communicate with Polinotes Discovery, to which the software must be registered in order to be tracked by client applications Polinotes Student.

The package "Service" contains the class TeacherServer, in which are implemented interfaces exposed as services Polinotes Teacher, this allows each instance of Polinotes Student to know the methods of subscription and proceed to download multimedia content within your presentation.

The others packages “Power Point Manager” and “Data” are not relevant for this section, but essentially they handle events and operations for extracting and manipulation of content from the current Power Point presentation to the instance of Polinotes Teacher being used when a professor.
6.1.5.3.  **WCF implementation on Polinotes Student**

The software architecture of Polinotes Student (see Figure 51) is composed by five different packages:

- Contracts;
- Services;
- One Note Manager;
- Data;
- GUI.

The package "Contracts" contains the interfaces defined as a service contract with the software communicates outside. Other interfaces in this package aren’t implemented and they are only established in order to communicate with other software package Polinotes.

The interface IStudentsInterfaceCallback contains functions that are all invoked from Polinotes Teacher and they basically allow it to send the contents of the presentation to Polinotes Student. It is the unique interface implemented in the package "Services" and the implementation in on the class Callbacks.

The method addText () sends the actual content of the application, the other functions, such as addHtml () and addMedia (), notify the client that there is a multimedia content to download, then Polinotes Student will carry out the download.

The stopClient () notifies the software that the server process running on the teacher's computer has stopped its operation and ends the session.

ITeacherInterface and IDataInterface are used to invoke functions on the instances of the software Polinotes Teacher and "DiscoveryServiceI" to communicate with Polinotes Discovery.

The package Services contains the class Callbacks which implements the functions that allows Polinotes Teacher to invoke and send content shown on the multimedia presentation to send to Polinotes Student.

The directory "Data" contains a single class "Connection" in which are stored all the connection parameters and active connections at the time.

The other packages One Note Manager and GUI contains classes that is used by Polinotes Student in the interoperability with Microsoft One Note and to build a graphical interface to support the student in the process of connection of this component with the instance of Polinotes Teacher being executed along the lecture.
In order to ensure two-way communication between applications Polinotes Teacher and Polinotes Student it is needed to implement a service contract that allows the process server, which resides on the professor's computer, to invoke functions on the client Polinotes Student.

The possibility of performing callback functions allows overcoming the classical concept of relationship between client and server. Typically, the client proceeds to the invocation of functions on the server and waits for a response; in this case it is possible to establish a relationship of equality between the two parties.

The use of callback methods allows the solution to make the most of the communication channel, as it gives the possibility to the Polinotes Teacher to send data only when it is needed without the need of client process Polinotes Student be forced to carry out periodic requests to the server. The communication between the applications is more efficient, as there is an exchange of messages only when a new multimedia slide is displayed.
6.1.5.4. WCF implementation on Polinotes Discovery

It is a very simple application and essentially consists of three different packages (see Figure 52):

- Contracts
- Services
- Data

In the package Contracts there is the interface DiscoveryServiceI that acts as a service contract and is exposed outside in order to provide the functions of recording a new teacher and request the IP address. The interface consists of three functions callable: for two instances of the software "Poli Teacher Notes" and reserved for the use of applications "Poly Student Notes."

The "subscribeTeacher ()" receives as input the name of the teacher and the IP address of the computer that is running "Poli Teacher Notes" and shall record the teacher in the list of active users. At the end of the lesson the software "Poli Teacher Notes" will remove your data from the service by calling the operation "unsubscribeTeacher ()".

The "getIpAddress ()" is reserved for instances of "Poli Student Notes" and receiving as input the name of the teacher returns the IP address of the computer to which the teacher is.

![Software architecture of Polinotes Discovery](image)

Figure 52 - Software architecture of Polinotes Discovery
The package Data contains the class "Teacher", which collects information on individual teachers registered for the service of discovery. The class "Teacher" associates for each teacher an identifier and the IP address he is using at that moment.

The package Services contains the class "DiscoveryService" that implements the interface "DiscoveryServiceI". It acts on the data during the recording of new teachers and search the IP address at the requests of the instances of Polinotes Student.

The Windows Communication Foundation "Poli Notes Discovery" exposed to the outside interface "DiscoveryServiceI."

### 6.2. Development of Polinotes Student App

This section describes the implementation of Polinotes Student App. It includes the main aspects of the development; presents how the design discussed along previous chapter and the integration with Polinotes Teacher and Polinotes Discovery are implemented.

#### 6.2.1. Objectives

The main goal of this applicative is to support the process of annotation for students along slides based lectures. For this purpose it aims to reduce the main constraints existing in traditional way of taking notes using pen and paper and enhance the annotation experience by using computational resources.

As discussed in previous chapters, an appropriate applicative to attend this task should be able to retrieve automatically content presented by a professor and make it available to students in an easy way to manipulate and changing that content.

In the other hand, independently from the lecture content received, students should be able to perform their annotations having the same facility as using a sheet of paper but offering features to overcome its main constraints such as manipulating digital elements or moving written content.

In order to support these objectives, this applicative has essentially to:

- Be integrated with WCF implementation of Polinotes in a way that allows an instance of the applicative to request IP of a professor computer on Polinotes Discovery, connects with Polinotes Teacher and receives all the textual content presented and performs requests for downloading images;
• Implement gestures and commands appropriated to carry out the process of annotation along the dynamic of a lecture by providing an appropriate user experience to students while handling this task.

• Follow design guidelines for Windows Store app in order to provide an immersive user experience for students, as expected from a Natural User Interface, and reduction of gaps of learning the software while interacting with multimedia content inside Windows 8 environment.

6.2.2. Technologies for development

Windows Store architecture

Windows Runtime (WinRT) is a platform that contains APIs used to build Windows Store applicatives. These APIs are available to developers in multiple languages, including JavaScript, C++, C#, and Visual Basic. The new Windows SDK for Windows Store applicatives also includes a subset of traditional Win32, COM, and .NET Framework APIs, as well as HTML5 and CSS3 APIs that are accessible to Windows Store application developers [7].
According to Microsoft [34], this API is expected to provide the same core services as the ones provided by Win32 but including also new resources to support communication between Windows Store Apps for sharing and retrieving information as well as features to manipulate audio and video content.

As shown in the figure 53, due to the API is placed on the top of the Windows core services it still provides to developers access to Windows 8 resources, including APIs for hardware such as built-in webcams, geolocation, light sensors and accelerometers. Moreover, fundamentals resources such as memory management, authentication, globalization and asynchronous processing.

**WCF on Windows Store App API**

Windows Store App provides to developers a subset of WCF functionalities to support communication and exchange of data between applicatives in its API. In this subset is also included the main resources and requirements used for Polinotes in order to establish communication with its components and broadcast multimedia content to clients, such as bindings and binding elements in its API. Therefore regarding this aspect there is no limitations for the applicative developed over this platform being integrated as a client in the package Polinotes [34].

**Development environment**

Microsoft Visual Studio 2012 is the software application that provides resources to developers for Windows Store apps. It has a collection of tools that supports the management of projects, writing the code, debugging capabilities and deployment of a Windows Store app.

For the development, the language chosen is C# mainly due Polinotes is also implemented with that language and so contributing for developing similar components for WCF communication as in Polinotes Students.

Associated to the C# development, the Windows Store App uses Extensible Application Markup Language (XAML) to define the user interface. XAML is a declarative language, it can initialize objects and set properties of objects, it uses a language structure that shows hierarchical relationships between multiple objects, and using a backing type convention that supports extension of types. Therefore it is used to create user interfaces elements such as controls, shapes, texts or other content present on the screen. XAML can be associated with a separate
code-behind written in C# that is able to respond to events and manipulate the objects that originally is declared in the XAML [7].

![Microsoft Visual Studio 2012](image)

**Figure 54 - Microsoft Visual Studio 2012**

### 6.2.3. Software architecture

The implementation of Polinotes Student App consists on the following five packages (see the UML diagram on Figure 55):

- Objects
- Components
- Polinotes Connection
- Canvas Manager
- GUI
Polinotes Connection

The package Polinotes Connection has the essential classes that allow the applicative to connect with Polinotes Teacher and Polinotes Discovery. It is very similar to the way Polinotes Student is implemented; inside this package it is possible to find three sub packages

- Contracts
- Data
- Services

As similarly described in Polinotes Student, the package "Contracts" contains the interfaces defined as a service contract. The interface IStudentsInterfaceCallback contains functions that are all invoked from Polinotes Teacher and they basically allow it to send the contents of the presentation to Polinotes Student. This interface is implemented in the package "Services" and the implementation in on the class Callbacks.

The others interfaces of package contracts, ITeacherInterface and IDataInterface, are respectively used to invoke functions on the instances of the software Polinotes Teacher and "DiscoveryServiceI" to communicate with Polinotes Discovery.

The package "Data" contains a single class "Connection" in which are stored all the connection parameters and active connections at the time.

Objects

The package “Objects” contains the implementation of the objects used along the process of annotation such as Images, Texts and Ink input; and also for objects created to support some gestures as the ones to create space and the rectangle that performs selection of multiple objects.

Another important object implementation within this package is the object to support multi-selection on a page, this implementation offers support to the user manipulate a group of objects and perform commands such as to replicate the objects chosen.

Components

This package implements classes that complement some functionality to objects. The class “ComponentCanvasContainer” is an implementation to support manipulations over a set of
objects. “ComponentColor” is the class used to handle the functionalities of changing color for diverse objects. “ComponentTextEdition” provides a set of resources to support the implementation of formatting text objects.

**Canvas Manager**

The package “Canvas Manager” contains classes that implement the concept of multiple pages to the annotation process. It handles all the operations of inserting, removing and browsing through different pages as well as manages new incoming content brought by Polinotes Teacher and process the logic of where and how to place this content in the applicative.

**GUI**

It is responsible for the visual elements of the applicative; the class MainPage is where the main controls for the canvas, navigation bar elements and app bar commands are declared. The class ConnectionUserControl contains the definitions for the interface that supports users during the process of connection of the applicative with Polinotes.
Design and implementation of a Natural User Interface to support the annotation process for multimedia learning

Diogo Borges Krobath

Figure 55 - Software architecture of the app
6.2.4. Implementation for Windows Store environment

Canvas

Canvas is the workplace where the most part of actions of annotation process are placed; the implementation consists on a representation of physical sheet of paper where students can manipulate diverse instances of the annotation elements brought by themselves or from Polinotes Teacher. An indication of the current page is shown to offer some orientation to the user.

![Figure 56 - Canvas implementation](image)

Navigation bar

This bar offers features to support the student while browsing along the pages containing notes (see Figure 57). The buttons Next and Previous are designed for browsing pages that are near to the current page. For browsing among more pages, the component slide bar provides a fast transition between a large set of pages and at the same time provides the visualization of the pages being browsed in the Canvas.
For the case the student wants to access a specific known page, the slide bar component offers the possibility to tap in a certain position to direct to the desirable page.

![Figure 57 - Navigation bar implementation](image)

**Command bar**

This bar provides the commanding operation for the applicative. On the bottom right of this bar it has icons to change interface state mode and to perform connection with an instance of Polinotes Teacher. The appearance of the bottom left side changes according to the interface mode.

When interface is in annotation mode and no element in the Canvas is selected, it displays commands to add elements such as images and texts to the annotation document.

For the case of an element is selected few more options are provided to student in order to issue an action into a group of elements. The actions implemented for this case are delete, duplicate and basic formatting.

Duplicate can be performed in two ways; by the operations of cut (moves the object to another page) or copy (moves an identical copy of the object to another page). For both an additional command “paste” is shown for the operation of placing in a desirable page the content intended to be duplicated or moved.

The basic formatting commanding implemented corresponds with functionalities to allow students to change the foreground color of texts and ink color of a handwritten input.

![Figure 58 - Command bar implementation](image)
6.2.5. Implementation of elements for the annotation process

Page

As mentioned before, this object is implemented over all the Canvas area and visually, as shown in the Figure 57, it reproduces the aspects of a physical sheet of paper.

Considering the scenario of a student following a lecture and receiving multimedia content from a Polinotes Teacher instance; the page is conceptually grouped according to the slide it received multimedia content from.

Whenever a new slide is introduced to students and its contents are ready to be broadcasted to students’ applications; the applicative creates automatically a new page to append the content to be received. In a situation where the content received exceeds the spatial capacity of a page, a new page is created automatically and the process of allocating content to users is switched to the new page.

Every time a new slide is introduced along the presentation, a new page is automatically created on the client side; therefore in principle the implementation doesn’t allow to students to have content of different slides on the same page.

The creation of new pages and allocation of multimedia content brought from the slides is performed in the background and it doesn’t change the focus of the current page where the student is interacting; therefore this procedure doesn’t disturb students when performing their annotations.

Image

Images are available to users in two ways; first is by directly adding through the commanding bar an image file from user’s computer or by receiving from Polinotes Teacher as part of the multimedia content presented on a slide.

The graphic representation of the image is inserted on the Canvas (see Figure 59) and over this representation it is fully implemented the gestures and commands discussed in previous chapters.

By the default an image is always placed after the last element found on a page; in case there is no further space in the page, a new page is automatically created and then the image is placed on that page. For the last situation, whenever an image is brought from a slide, the process of
allocating the image in the next page is performed on the background so aiming to not disturb any annotation procedure the student might be performing at the same time. When an image is added by commanding, it is assumed that the next interactions performed by the user will be over that image; therefore in this situation the new page just created is then changed to the current page of manipulation and allowing the user to interact with the image just inserted.

Pen handwritten

The implementation of pen handwritten was performed within the framework for Pointer events of WinRT API (this framework is explained in details in the next section). When a student touches the pen on the screen and continues moving it a sequence of continuous lines are appended into Canvas and this process keeps on until the user lifts the pen from the screen contact.

An issue for this approach is determining which lines composes a word in order to offer later functionalities for allowing users to move words or even change its ink color.

Diverse techniques can be applied to solve this problem. The solution adopted for this project consists on a timer trick to join blocks of words that makes sense to be together in the context.
This trick essentially adds to the same group of words all the ink produced when a pen keeps moving along the screen and as soon as user lifts the pen it is counted a few seconds period; in case there is another interaction within this period then the incoming ink is considered as part of previous group; otherwise, when the interaction is performed after the period, a new group is created and further interactions is considered as part of the new group.

Figure 60 demonstrates an example where this technique was applied, it can be observed that the black text isn’t continuously connect but since all the lines were drawn inside a specific time they are considered as belonging to a single group. The red group, that was initially black, was started to be drawn after the threshold time, so it is considered a new group of text. In the example an operation of changing color was done in order to highlight the differences of the groups.

Figure 60 - Two groups of handwriting texts defined (red and black) by timer trick

Another simple technique applied in this element is based on comparing the bounding boxes involving groups of handwritten inks.
The calculation of a bounding box is performed by firstly identifying extreme points for the sides up, down, left and right on a group, then defining the rectangle in which the sides adjusts to those points.

After a group of words is just defined by timer trick technique, in other words, when there is no pen interaction with the screen within the minimum period of time established; it is then calculated a bounding box for the group just created. In case the bounding box can be contained inside any other bounding box of previous groups, the group just created is then merged to the group that contains its bounding boxes.

This technique turns out to be useful for the cases when students are revising his notes and desires to add accentuation signals to the words; according to the technique just presented the accentuation mark should be contained on the bounding box of group of ink representing the work and therefore it is merged to the word.

Figure 61 demonstrates the result of this technique, on the left side the word doesn’t have the tittle mark over the character “i”, after a while the tittle mark is placed and due the technique that mark is merged to the group composing the word. The selection of the word, as shown on the right side comprises the mark just added.

![Figure 61 - Bounding box trick with tittle mark](image)

**Text**

The implementation of this object can be visualized in the figure 62. When in edition mode, it offers tools to perform some basic formatting such as applying bold, italic, underline, change font color and highlight. Besides that this object offers the virtual keyboard to support the input of texts.
The edition mode also provides the “redo” and “undo” commands. These features are helpful when user wants to return quicker to a previous object state of the text. It turns out to be an interesting mechanism for the annotation process due the level of accuracy and time demanded to execute simple operations such as selection fragment of text and then in sequence the operation to be applied. Therefore this mechanism allows users to perform transitions between diverse sates of formatting and content of text in an efficient manner.

As well as images, object texts are also touch interactive and offer the same basic operations such as moving, resizing and rotating when user interacts with them.

The logic of creation of these objects follows the same rules of images. They can be appended to the notes by commanding, from an icon on the command bar, and also brought from the content of a slide presented in class, through Polinotes Teacher.

The allocation of these objects is also similar, they are always placed after the last object found on the page. For the case when there is no space on the page, a new page is then created and the object just created is allocated in this new page. Still considering the case when a new page is created, if the object text comes from a slide, the process of allocating the new object in the new page is performed in the background and always the object coming from this modality is created in manipulation mode. However when the object is created by using the commanding bar, the page is automatically changed to the page just created and the new object text will be in edition mode (with the virtual keyboard support) in order to receive the text input desirable for the student.
6.2.6. Implementation of gestures

The Windows Runtime platform APIs enables applicatives to detect and respond across different input modes and devices such as pen, touch and mouse; as well as supports gestures range from simple interactions like tapping to more complicated manipulations like zooming, panning, and rotating. The API handles user interactions through three types of interaction events: pointer, static gesture, and manipulation.

- **Pointer events**: are used to get basic contact info such as location and device type, extended info such as pressure and contact geometry, and to support more complex interactions. Pointer events provide a unified and streamlined way to get mouse, touch, and pen data. They are intended for scenarios where there are interest of retrieving multiple pointers and their relationships, or when there is a need to examine specifics of each pointer such as exact coordinate position.

- **Static gesture events**: are used to handle static single-finger interactions such as tapping and press-and-hold (double-tap and right-tap are derived from these basic gestures). Static gesture events are triggered after an interaction is complete.

- **Manipulation events**: are used for dynamic multi-touch interactions such as pinching and stretching, and interactions that use inertia and velocity data such as panning/scrolling, zooming, and rotating. Manipulation gesture events indicate an ongoing interaction. Manipulation gesture events start firing when the user touches the element and continue until the user lifts the finger or the manipulation is canceled. The info provided by the manipulation events doesn't identify the interaction. It specifies input data such as position, translation delta, and velocity. This data is then used to determine the manipulation and perform the interaction [34].

More precisely these interaction events are handled through the events labeled on the Table 15.

The following subsections offers details of the implementation of the set of the gestures defined in the previous chapter. Some of these of events described in this section are worthy to be understood since they influence the way gestures were implemented on the applicative. In the following are also described those more relevant manipulation and point types events and the how they participate in the dynamic of a gesture implementation.
Table 15 - WinRT API events

<table>
<thead>
<tr>
<th>Type</th>
<th>WinRT API events generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer events</td>
<td>PointerCanceled PointerCaptureLost PointerEntered PointerExited PointerMoved PointerPressed PointerReleased PointerWheelChanged</td>
</tr>
<tr>
<td>Static gesture events</td>
<td>Holding Tapped DoubleTapped RightTapped</td>
</tr>
<tr>
<td>Manipulation events</td>
<td>ManipulationCompleted ManipulationDelta ManipulationInertiaStarting ManipulationStarted ManipulationStarting</td>
</tr>
</tbody>
</table>

6.2.6.1. Manipulation events

The implementation of these gestures using the manipulation events framework provides useful properties for the process of annotation. Their dynamic involve the negation operation; when a user start an action he can return to the previous state by doing the opposite action; for example executing a rotation clockwise turns an object but if the same amount of rotation is performed anticlockwise the object returns to the original position.

The user is also able to perform at the same interaction with an object the execution of two or more gestures in parallel, for example rotating and increasing the size of an object when moving away and spinning at the same time fingers.

The commutative property is also preserved among these gestures; when user changes the order of operations and the result is still the same, resuming to the previous example in case the student performs rotation and then in sequence resizing he achieves a result that can also be reached by executing resizing and rotation of course with the same correspondent parameters for both as applied before.

The most relevant manipulation events for this project are:
• **ManipulationStarting**: this event fires basically any time that a pointer interacts with an element where the element enables manipulation. This includes cases where the manipulation is interpreted as a gesture and the pointers never move, for example a Tapped or Holding gesture.

• **ManipulationStarted**: it represents that the manipulation recognition logic has detected pointer movement. In this case it's the second manipulation event to fire in a typical sequence, firing after ManipulationStarting.

• **ManipulationDelta**: this event occurs as many times the input device changes position during a manipulation.

• **ManipulationInertiaStarting**: event occurs when the input device loses contact with the object during a manipulation and inertia begins. This event enables developers to specify the deceleration of the manipulations during inertia. This is so your object can emulate different physical spaces or attributes if you choose. For example, suppose your application has two objects that represent items in the physical world, and one is heavier than the other. You can make the heavier object decelerate faster than the lighter object.

• **ManipulationCompleted**: event occurs when the manipulation and any inertia ends. That is, after all the ManipulationDelta events occur, the ManipulationCompleted event occurs to signal that the manipulation is complete.

The following gestures were implemented under manipulation event framework and they use the events just described.

**Rotate**

It uses manipulation event data available on ManipulationDelta in order to identify the angle of rotation performed by two fingers over an object and apply the visual transformation to the correspondent object Rotation. It is handled differently for object Pages, for this specific case it responds to hardware events when changing the positioning.
Resize

The manipulation event data offers information about the changes of distance between the two fingers over the objects; it uses this information to perform proportional variations of the size of the object.

Move

Uses manipulation event data in order to identify the translation performed by one finger over the object; the event handling use this information to determine the new positioning and offer the feedback of the object following.

Change distance among objects

The implementation is based on the gesture Move and its essence can be decomposed in diverse gestures Move being executed with different fingers performing some movement over different objects

6.2.6.2. Pointer events

This set of events implements gestures that happens over more specific contexts and demands access to more detailed data in different moments of the interaction. Some of the gestures implemented for this gesture framework don’t have all the features presented for Manipulation interaction events such as negation operations to return to a previous state. This case happens for the “Selection inside rectangle” gesture; after performing the gesture by lifting the finger there is no “Unselecting inside rectangle”.

However, still through reciprocal operations is possible to return to a state similar to the one found before user interaction; following the previous example with “Selection inside rectangle”, if the user double taps the Canvas it unselects the elements and therefore returns to previous state before interaction.

Although for some cases it is achieved, this set of gestures are not intended to support commutative operations or even parallel operations as the framework for manipulation events provides.

The most relevant pointer events for this project are:
• **PointerEntered**: this event fires in response to a pointer moving into the element's bounding area. Touch, mouse, and pen/stylus interactions are received, processed, and managed as pointer input in Windows Store apps.

• **PointerMoved**: occurs when a pointer moves while the pointer remains within the hit test area of this element.

• **PointerReleased**: occurs when the touch that previously initiated a Press action is released, while within this element. It is noteworthy that the end of a Press action is not guaranteed to fire a PointerReleased event.

• **PointerExited**: event fires in response to a pointer that was initially in the element's bounding area leaving that bounding area.

The following gestures were implemented under pointer events framework and they use the events just described.

### Next/previous document

The implementation consists in identifying one finger down when interface is in visualization mode and according with data movement provided by this gesture pattern along horizontal direction it recognizes as next or previous gesture.

### Selection inside rectangle

Identifies one finger down when interface is in annotation mode, the initial position of finger corresponds to one of the vertex of the rectangle. As long as user moves his finger along the Canvas a rectangle is built and its opposite vertex (using as referential the vertex placed on the starting position) follows the movement of the finger. At the position where the user lift his finger from the screen it is considered as the final position of opposite vertex; therefore the final shape of the rectangle.

The objects contained within this rectangle are shown as selected and available for group commanding operations. The figure 63 demonstrates it, the gray rectangle represents the rectangle built through touch manipulation on the Canvas and the blue rectangles indicate the objects selected by this procedure.
Create/reduce space

Its implementation is based on identifying two fingers contact over Canvas area while interface is in annotation mode. After recognition of those fingers, it is tracked the vertical movement of the fingers; when they are moving away the gesture “create space” is performed and on the other way, if they are moving closer the gesture “reduce space” is triggered.

Associated with the dynamic of both gestures, elements around them should follow the logic of the operation and also perform automatically operation of moving vertically up or down according to the gesture.

The movement of objects around is essential to provide feedback to user and effectively achieve what is intended which is generate space or reduce in a determinate region in order to support in sequence other operations supporting the annotation.

The logic of moving objects has also to consider situations such as the boundaries of the Canvas or the current page. This situation appears when “create space” is performed and the implementation handles the top boundary of the page as an obstacle not surmountable while on the bottom boundary a more complex reasoning is placed.

When an object reaches the bottom boundary, it is analyzed the content of the next page. In case there is no next page for current slide number, a new page is automatically created and the object is moved over there. If there is a next page for current slide and that page has enough space on the top comparing object dimensions, the object then is moved. On the other hand for
the case there is no space at the top for the object, the object movement is blocked on the bottom of the page.

![Image](image.png)

Figure 64 - Gesture create space

Select part of content

This gesture is already implemented by the native component responsible for input text (RichEditBox component) on the object text. The dynamic of this gesture is based on holding a finger down on a certain position of a text and then moving from this position to the final position of the fragment. The component provides a circle object to receive the finger input when it is about to start and also visual feedback while the execution by highlighting the fragment being selected by the gesture.

6.2.6.3. Static gesture event

This set of gestures is the simplest and easiest one to be performed and due its simplicity the framework of events to handle them doesn’t offer much more data besides where and when they happened.

Considering these circumstances, they are related with operations that changes states of the interface due these operations have a nature of demanding be executed in a short period of time and are not tolerant to false positives.

As shown in the table 16, basically two events were used to implement the gestures needed: Tapped and Double Tapped.
Design and implementation of a Natural User Interface to support the annotation process for multimedia learning

Diogo Borges Krobath

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Elements that handle static event</th>
<th>Static gesture event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tap to execute</td>
<td>An icon on app bar representing a command</td>
<td>Tapped</td>
</tr>
<tr>
<td>Tap to select an object</td>
<td>Any object of the annotation process placed over the Canvas</td>
<td>Tapped</td>
</tr>
<tr>
<td>Change to edition mode</td>
<td>Text object on manipulation mode</td>
<td>DoubleTapped</td>
</tr>
<tr>
<td>Change to manipulation mode</td>
<td>Any place over Canvas</td>
<td>DoubleTapped</td>
</tr>
</tbody>
</table>

Table 16 - Static gestures implemented
7. Conclusions and future work

This chapter presents the main conclusions of this thesis work and based on the results and analysis performed, suggests directions to be followed by future works.

7.1. Conclusions

This work provided a prototype (Polinotes Student App) able to handle student’s annotations along a lecture and due the integration implemented with the Polinotes package, receive multimedia content presented by a professor through Polinotes Teacher and join these content to the notes being performed by a student.

The design of this interface was based on multimedia learning principles, as well as the most relevant needs of features identified on current methods students use for notes taking during lectures and later alignment with platforms such as Windows Store App on Windows 8 environment to support interactivity in diverse modalities (multi-touch, pen, mouse and keyboard) and the immersive interface paradigm of NUI.

Comparing the results of this work with a typical software for taking notes that relies on a GUI paradigm, such as Polinotes Student which was based on Microsoft OneNote 2010 interface, it is evident the benefits of having an interface totally designed for the annotation process instead of using a software more oriented to “document edition” concept such as Microsoft OneNote, the applicative of this work provides a more intuitive and simplified set of functions; in which focus on tasks considered as the core for carrying the annotation process such as inserting annotation objects, moving objects, changing content and navigation within digital notes.

Once the hardware capabilities evolves and offers even thinner tablets and closer to the way people manipulates a sheet; the NUI design approach for multi-touch and pen of this work could offer similar user experience to the users handling a traditional sheet and pen while doing annotations; which is extremely beneficial for getting the user familiar with the software.

The problem of changing interface modes, found on most software solutions the students use to perform their annotations, was solved by essentially dividing the modes into the main functions the annotation: encoding function and storage function. This division represents what this work
believes to be the most appropriate for minimizing the switching between modes and therefore reducing efforts while using the applicative.

Despite these functionalities, it is relevant to mention the development of this applicative was carried out as a Windows Store App, so enabling this software to be easily integrated with Polinotes package by being under Windows environment as well as providing to the users the most recent outcome from Microsoft in terms of human user interaction for multi-touch devices that is the Windows 8 which even regarding its controversial acceptance and design issues, it will probably be the basis of next releases of that operational system; therefore this approach is aligned with current trends for incorporating even more the principles of Natural User Interfaces to users common tasks.

The benefits of using such approach for education are still unclear and still not deeply explored by researchers, however this thesis work presents some cognitive aspects for learning that support this approach and claims that it is appropriated to motivate students to be actively engaged with a lecture and be immersed into the task of taking notes.

### 7.2. Future work

In terms of functionalities for the applicative, future work could include resources of OCR recognition for transforming the handwritten input into a traditional format in order to facilitate edition and formatting on the text operations as well as improve the clarity of notes therefore enhancing the reading process when reviewing the notes.

Furthermore, adding mechanisms for persisting and retrieving the notes documents would be essential to the storage function of the annotation. In order to facilitate the study, the retrieving process should consider the association of the notes document to the original slide from where a note was based on.

Another useful functionality to support the annotation is the possibility to perform “undo” and “redo” operations for the actions over annotation objects done on the Canvas. It would provide reciprocal operations to support students to retrieve quickly a desirable state of the interface therefore improving the dynamics of the annotation.

Moreover, since the guidelines for the design were driven by only considering the learning involved when a student receives information from the multimedia content presented as a slide by a professor; further approaches exploring deeply how the dynamics of student-student and student-professor works and contributes for learning might be useful for incorporating functionalities to the applicative.
Resources for enabling audio or even a video recording of a professor explanation could be incorporated to the multimedia content broadcasted to students. For this case students would have not only their notes followed by the images and texts of the Power Point slide but also the fragment of audio or video containing the original explanation of professor at the time a specific slide were presented along the lecture which would indeed facilitate the study after a lecture.

Despite of the voice and image of a professor, mechanisms to retrieve any handwriting input used by them in order to explain a specific content, commonly used when a professor wants to stress or add information not in the slides could allow, and send this content to students could even suppress the need for a physical classroom and reaches a level in which it is feasible virtual classrooms where by using a hardware such as a Table-PC a student could connect to a lecture and receive fully all the content and explanations of a professors in a way that facilitates his annotation and study.

Still considering the connectivity between students with professors; a new channel of information flow could be thought for future work; in which students don’t only receive the multimedia content of the class but are also able to actively send information to professor or even have the control of the class presentation and the content shown to other students for a period defined by professor.

Such channel could be object of a more quantitative research investigating the way students are active in a lecture and responds to the multimedia content presented along it; by providing to professors mass data about the interactivity of students while using the applicative to take notes could offer real time feedback, for example, about how many students are taking notes, indicators about how should be the pace of the presentation or even obtain collective feedback about students interest on the lecture through interactive pools.

Summarizing, besides the improvements on the basic functionalities of the applicative for the annotation process, future work based on this thesis could also be oriented by principles to foster interactivity among students and professors during a lecture as well as enhance the collaboration of diverse students at the same time with the content dealt in class and interfaces able to immerse and offer resources to students in order to suppress context switching events.
References


[24] Margaret S. Chan, John B. Black, “Direct-manipulation animation: incorporating the haptic channel in the learning process to support middle school students in science learning and mental model acquisition”, Columbia University


